uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Electrónica Industrial y Automática

2017 - 2018

Trabajo Fin de Grado

Controller For Automated Navigation Of A Differential Drive Mobile Robot

---

Álvaro Gil Cabezas

Tutor: Abdulla Hussein Abdulrahman Al Kaff

Departamento: Ingeniería de Sistemas y Automática

Leganés, septiembre de 2018

AGRADECIMIENTOS

Para comenzar, he de mencionar a mi familia, ya que no solamente durante el periodo de desarrollo de este trabajo fin de grado, si no durante los cuatro años de grado ha estado siempre ahí, apoyándome día a día y ofreciéndome su ayuda y consejos siempre que fue necesario. Gracias. A mis amigos, especialmente a quienes han compartido esta experiencia conmigo, gracias. Ha sido muy importante para mí la comprensión y el consuelo mutuo ofrecido. Por último, y no por ello menos importante, gracias a todo el departamento, en especial a mi tutor, Abdulla, gracias por tu ayuda y tus consejos, sin ti esto no habría sido posible. Gracias.

RESUMEN

En un creciente mercado de vehículos autónomos, MINESWEEPERS, Towards a Landmine-free world, propone un enfoque diferente para los mismos, la búsqueda de minas terrestres antipersona. Para ello ha propuesto una competición a la cual se ha adherido la Universidad Carlos III de Madrid.

En el presente trabajo, se ha desarrollado un algoritmo de control basado en control óptimo, que permitirá a un UGV de tamaño reducido moverse de manera autónoma en un entorno controlado y desconocido, dividido en celdas y con una extensión de 400m^2. Se ha llevado a cabo una programación de este en lenguaje C++, haciendo uso del sistema operativo ROS, así como de librerías tales como MAVROS. Esto permitirá al UGV intercambiar información entre los diferentes componentes internos tales como el GPS o el acelerómetro, además de con otros sistemas externos, que, pese a no desarrollarse en este trabajo, guardan relación con el mismo y son necesarios para la competición.

A lo largo del trabajo se expondrá y justificará la elección de los componentes de hardware incluidos en el sistema, el desarrollo del algoritmo de control, su programación y la integración del sistema completo que participará en la competición. Posteriormente se harán pruebas que den validez al controlador. Se terminará con las conclusiones obtenidas del proceso de desarrollo e implementación de un controlador para un vehículo autónomo y posibles mejoras que puedan añadirse en el futuro.

Palabras clave: autónomo, control óptimo, ROS, UGV.

ABSTRACT

In a constantly growing autonomous vehicle market, MINESWEEPERS, Towards a Landmine-free world, proposes a different approach for them, the search for antipersonnel landmines. To this end, they have proposed a competition to which the University Carlos III of Madrid has acceded.

In the present project, a control algorithm based in optimal control has been developed. It will command a small UGV for moving autonomously in a controlled and unknown field, divided in cells and with a total area of 400 m^2. The mentioned controller has been programmed using the language C++, alongside ROS and some of its libraries such as MAVROS. This will allow the UGV to exchange data between all its components such as the GPS or the accelerometer. These libraries will also be used to communicate with external systems, that, although they are not developed in this project, are related to it and are necessary for the competition.

Throughout the project, the hardware components choice, the development of the control algorithm, its programming and its full integration in the complete working system will be explained and justified. Later, the controller will be fully tested. The project will end with the conclusions extracted from the control algorithm development and implementing process and some possible improvements that could be added to it in the future.

Keywords: autonomous, optimal control, ROS, UGV.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTADO DE ACRÓNIMOS

UGV	Unmanned Ground Vehicle (Vehículo Terrestre No Tripulado)
UAV	Unmanned Aircraft Vehicle (Vehículo Aéreo No Tripulado)
GPS	Global Positioning System (Sistema De Posicionamiento Global)
ROS	Robot Operating System
Controlador PID	Controlador Proporcional-Integral-Derivativo
RFID	Radio Frequency IDentification (IDentificación Por Radio Frecuencia)
UWB	Ultra Wide Band (Banda Ultra Ancha)
IR	Infrared (Infrarrojos)
SO	Sistema Operativo
ros-pkg	ROS Package (Paquete De ROS)
FCU	Flight Control Unit (Unidad De Control De Vuelo)
SCB	Single Board Computer (Ordenador De Placa Reducida)
DC	Direct Current (Corriente Continua)
QGC	QGroundControl
Minas AP	Minas antipersona
ESC	Electronic Speed Controller (Variador)
BEC	Battery Eliminator Circuit (Circuito Para Controlar Batería)
CPU	Central Processing Unit (Unidad Central De Procesamiento)
LI-ion	Litio-Ion
LiPo	Polímero de litio
NiCd	Níquel-Cadmio
NiMH	Níquel-Metal Hidruro

NiFe	Níquel-Hierro
GPU	Graphics Processing Unit (Unidad De Procesamiento De Gráficos)
S.I.	Sistema Internacional
AP	Access Point (Punto de Acceso)
SSH	Secure Shell

1. INTRODUCCIÓN

La elevada mortalidad que suponen las más de 110 millones de minas enterradas que se encuentran localizadas en zonas de guerra, ha impulsado a la concienciación y a la promoción de soluciones que permitan desactivar y retirar las mismas, sin poner en riesgo vidas humanas en el proceso.

Esto ha llevado a organizaciones tales como *MINESWEEPERS, Towards a Landmine-free world*[1] a realizar competiciones internacionales de manera anual, en diferentes partes del mundo para promover soluciones autónomas para la detección de las minas antipersona, tanto para aquellas que se encuentran enterradas como para las superficiales.

La Universidad Carlos III de Madrid se ha adherido a la competición, promoviendo una serie de trabajos de fin de grado y de máster, entre los que se encuentra el presente. El vehículo utilizado para la detección de minas en la competición es un rover, cuyo movimiento se realizará de manera autónoma gracias al controlador desarrollado en el trabajo, siguiendo las normas del concurso para cumplir el objetivo de encontrar y clasificar todas las minas antipersona localizadas en el terreno.

Fig. 1.1. UGV Rover Wild Thumper [1]

En el presente trabajo, partiendo del estudio realizado sobre el estado del arte en control de motores y vehículos eléctricos utilizando diferentes tipos de control tales como PID, se desarrollará un algoritmo de control basado en control óptimo, que permitirá al rover moverse a lo largo del terreno a inspeccionar en busca de minas.

[1] http://www.landminefree.org

1.1. Problema

El desarrollo de un controlador para un vehículo autónomo presenta una serie de dificultades, cuya importancia varía según la aplicación que vaya a tener el mismo. En este caso, el principal desafío que hay que solventar es el de llevar a cabo un movimiento fluido y preciso en un terreno desconocido a priori además de irregular.

Dado que se realizarán múltiples movimientos rectilíneos en terrenos con desniveles y que carecen de ningún tipo de pavimentación, si no se implementa una solución de control muy precisa, se acabaría acumulando un error gradual que desembocaría en unos resultados no satisfactorios y potencialmente perjudiciales, tanto para la competición como para futuras aplicaciones del modelo en diferentes entornos.

Añadido a esto, será necesario implementar un sistema de comunicación robusto y fiable, para la interacción entre los diferentes sistemas involucrados en la detección de minas. Dichos sistemas son la visión por computador, el detector de metales, la transmisión de la información de la localización de las minas y el movimiento del rover, siendo este último el que tendrá lugar en el trabajo. Gracias a la comunicación, cada uno de los elementos del sistema tendrá un funcionamiento dependiente de los demás en muchos aspectos sin implicar esto que no sea seguro o confiable.

Por último, dado que hay involucrados numerosos sensores y actuadores, cada uno de ellos con componentes electrónicos y mecánicos, todos ellos susceptibles a fallos y roturas, que pueden afectar directa o indirectamente al funcionamiento del sistema, es indispensable monitorizarlos. Uno de los elementos que es indispensable monitorizar, tanto su carga como su salud general, es la batería, ya que cualquier tipo de fallo puede causar accidentes. Además, será necesario comprobar que todos los componentes utilizados envíen y reciban datos de manera correcta, lo cual será un claro indicativo de un correcto funcionamiento de cada uno de ellos.

1.2. Solución propuesta

Con el objetivo de conseguir un control preciso, y minimizar los errores, se ha implementado un método de control alternativo al PID, y que requiere de una menor capacidad computacional para llevarse a cabo, siendo más preciso. Dicho método es conocido como control óptimo, que pese a ser menos utilizado, posee ciertas características que son de gran utilidad en este proyecto.

El control óptimo destaca por la posibilidad de definir de manera concisa las condiciones de los puntos de inicio y fin de las trayectorias, y de la ejecución de estas. Además, llevando a cabo unos cálculos previos extensos y precisos, es posible minimizar errores en la realización de los movimientos.

También, puesto que el presente trabajo es una fracción de un proyecto colaborativo mayor, que requiere de sistemas de comunicación fiables y complejos, todo el proyecto se ha llevado a cabo en la plataforma R.O.S. (Robot Operating System). Se trata de un framework de código abierto diseñado para facilitar a los desarrolladores crear software para cualquier tipo de robots. R.O.S. permite tanto la comunicación interna de los distintos sistemas del rover, como una comunicación externa con otros elementos necesarios para la competición.

Para finalizar, es necesario apuntar que mediante R.O.S. se llevará a cabo la supervisión de todos los componentes. Tanto las baterías que alimentan todo el sistema, como los distintos actuadores mecánicos, son los componentes a los que debe prestarse una mayor atención. Por un lado, se comprueba que las baterías tengan una salud correcta, esto es, que su carga máxima no sea inferior 80% de su capacidad, y que su descarga sea correcta, de forma que no puedan drenarse por problemas como cortocircuitos y causar accidentes como explosiones o igniciones. Además, se comprueba periódicamente que los motores del vehículo funcionan correctamente, ya que pueden obstruirse debido a la suciedad presente en el suelo en el que se están moviendo, pudiendo causar fallos en la ejecución de las trayectorias previstas.

1.3. Planificación

Con el objetivo de plasmar los periodos que abarca cada proceso llevado a cabo en el desarrollo del presente trabajo, se ha realizado un diagrama de Gantt (ver Tabla 1). Este diagrama permitirá una comprensión más sencilla de los pasos llevados a cabo en la realización del trabajo, así como del tiempo invertido en cada uno de ellos.

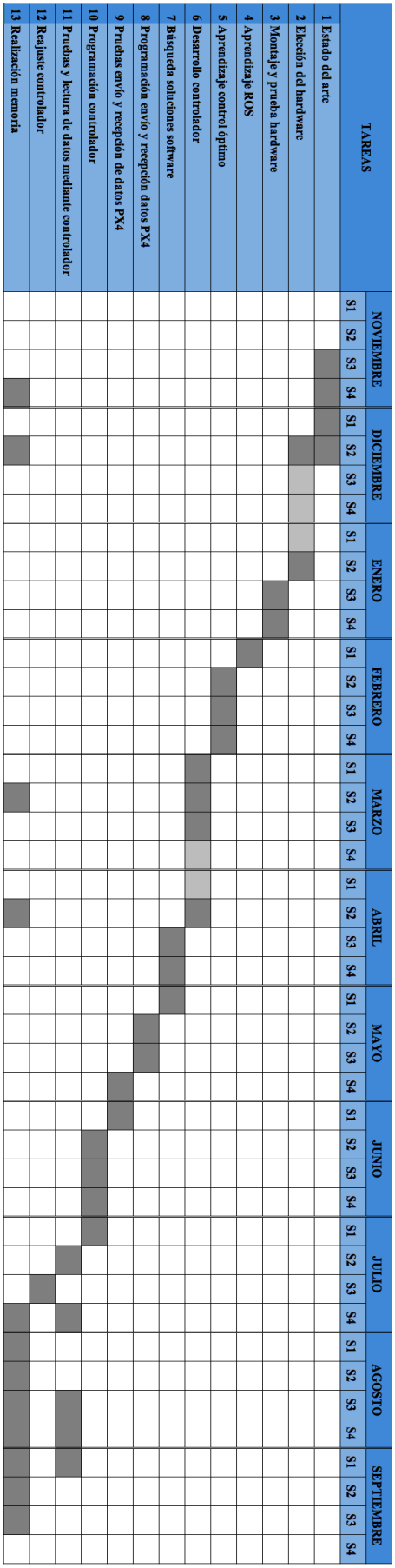

Tabla 1. Diagrama de Gantt de las tareas realizadas del trabajo

1.4. Presupuesto

En el apartado actual, se detallará un presupuesto aproximado del coste que conlleva este trabajo, en el se incluyen tanto la mano de obra, según las horas invertidas y el precio de esta, así como el coste de los elementos necesarios para su desarrollo e implementación.

Para obtener una medida aproximada de las horas invertidas en la realización del trabajo, se han extrapolado los datos del diagrama de Gantt mostrado en el epígrafe anterior. Además, acorde con la oferta y demanda de ingenieros que acaban de terminar los estudios en el mercado actual, específicamente programadores, puede aproximarse el coste de cada hora de trabajo en unos 15€, salario neto. Puede considerarse el trabajo realizado en aproximadamente media jornada laboral, unas 4 horas diarias, durante 9 meses, por tanto, obtenemos 800 horas invertidas en la realización del trabajo.

CÓDIGO	UNIDAD	DESCRIPCIÓN	MEDICIÓN	PRECIO UNITARIO	PRECIO TOTAL
		CAPÍTULO 1. EQUIPO			
1.01	Ud.	**Rover Wild Thumper 6WD** Suministro de vehículo terrestre Dagu Rover Wild Thumper 6WD. Con seis motores 7,2Vy realción de tranmisión 34:1 y seis ruedas 120mm cada una. Chasis negro.	1	250,00	250,00
1.02	Ud.	**Pixhawk 2** Suministro Sistema de autopiloto Pixhawk 2. 32bit STM32F427 Cortex-M4F® core con FPU a 168 MHz / 252 MIPS, 256 KB RAM, 2 MB Flash (totalmente accesible), 32 bit STM32F103 failsafe co-procesador y 14 salidas PWM / Servo.	1	250,00	250,00
1.03	Ud.	**GPS Here+ V2 RTK GNSS(M8P)** Suministro GPS Here+ GNSS para Pixhawk 2. Compatible con Galileo, GLONASS y BeiDou. Con seguridad y protección de integridad. Detección avanzada de falsificación de posición. Sensibilidad de -157dBm. Incluido sistema RTK para rover.	1	60,000	600,00

CÓDIGO	UNIDAD	DESCRIPCIÓN	MEDICIÓN	PRECIO UNITARIO	PRECIO TOTAL
1.04	Ud.	**Batería XXL Power 2600mAh 30C 11,1V** Suministro de la batería XXL Power 2600mAh. 30C 11,1V. Con una capacidad típica de 2200mAh. Formada por celdas de Polímero de litio. Voltaje de 11,1V. Tres celdas (3S). Dimensiones de 105 x 34 x 23.3 mm. Tasa de descarga típica de 30C (150A).	1	17,00	17,00
1.05	Ud.	**Batería Desire power V8 Series 3s 5000mAh 30C** Suministro de la batería de polímeros de litio Desire power V8 Series 3s 5000mAh 30C. Con una capacidad típica 1500mAh. Voltaje de 11,1V. Dimensiones de 29.5x45x136mm. Peso de 374g. Tasa de descarga típica de 30C (150A), y máxima de 60C (300A).	1	44,50	44,50
1.06	Ud.	**ESC HobbyWing QUICKRUN 1060 Brushed** Suministro ESC HobbyWing QUICKRUN 1060 Brushed. Resistente al polvo y al agua. Corriente marcha adelante 60A (Pico de corriente 360A). Corriente de marcha atrás 30ª (Pico de corriente 180A). Rango de voltaje 2-3S en baterías LiPo o 5-9 en baterías NiMH. Tamaño del motor 12RPM o mayor. Resistencia desvío 0,001Ω**.** BEC salida 2A/5V. Frecuencia PWM 1KHz. Tamaño 36,5 mm x 32 mm x 18 mm**.** Peso de 39g. Modos adelante/atrás/freno.	2	26,00	52,00
1.07	Ud.	**Odroid XU4** Suministro ordenador Odroid XU4. CPU Samsung Exynos5422, GPU Mali-T628 MP6, 2GB LPDDR3 RAM PoP, almacenamiento eMMC5.0 HS400 Flash, dos USB 3.0, un			

CÓDIGO	UNIDAD	DESCRIPCIÓN	MEDICIÓN	PRECIO UNITARIO	PRECIO TOTAL
		USB 2.0, un puerto Gigabit y un puerto HDMI 1.4a.	1	60,00	60,00
1.08	Ud.	**Transmisor y receptor RC Turnigy 9X** Suministro control remoto RC Turnigy 9X. 9 canales de comunicación a 2,4GHz y pantalla LCD. Incluye transmisor Turnigy 9x 9Ch, modulo Turnigy RF9X-V2 y receptor Turnigy 9X8C-V2	1	65,00	65,00
1.08	Ud.	**Ordenador portátil** Suministro ordenador portátil capaz de ejecutar el software Ubuntu 16.04 y ROS. Con una tarjeta de red capaz de comunicarse mediante protocolos SSH.	1	600,00	600,00
		CAPÍTULO 2. SOFTWARE			
2.01	Ud.	**ROS** Licencia uso ROS.	1	0,00	0,00
2.02	Ud.	**Ubuntu 16.04** Licencia uso Ubuntu 16.04.	1	0,00	0,00
		CAPÍTULO 3. Recursos Humanos			
3.01	H.	**Ingeniero Técnico Industrial, especialidad en Electrónica Industrial** Coste del trabajo llevado a cabo durante el periodo indicado. Precio unitario expresado en €/h.	360	15,00	5.400,00

Tabla 2. Presupuesto del trabajo

CAPÍTULO	RESUMEN	EUROS	%
1	Equipo……..	1.938,50	26,40
2	Software……..	0,00	0,00
3	Recursos Humanos……..	5.400,00	73,40
	TOTAL PRESUPUESTO GENERAL	7.338,50	

Asciende el presupuesto general a la expresa cantidad de **SIETEMIL TRESCIENTOS TREINTA Y OCHO EUROS con CINCUENTA céntimos.**

Tabla 3. Resumen del presupuesto del trabajo

1.5. Normativa

A fecha de septiembre de 2018, España carece de leyes referentes a los vehículos autónomos no comerciales. Las leyes nacionales para coches autónomos únicamente hacen referencia a los vehículos conducibles, y afirman que únicamente pueden ser probados por instituciones y empresas, en vías destinadas a este fin. Dado que el vehículo utilizado no se enmarca en estos parámetros, puede afirmarse que no se encuentra regulado legalmente.

Sien embargo, puesto que el trabajo se ha realizado en el marco de una competición, y los elementos utilizados no se encuentran regulados por las leyes locales, puede concluirse que el presente trabajo no se encuentra sujeto a ninguna normativa legal, a excepción de las condiciones impuestas por la organización de la competición. Dichas normas se desarrollarán posteriormente en el trabajo.

1.6. Estructura del trabajo

El presente trabajo está estructurado en capítulos, en cada uno de ellos se ha realizado una tarea relacionada con el objetivo del sistema diferente. La división realizada es la siguiente:

- Un primer capítulo, el presente, en el que se realiza una introducción del trabajo, con el objetivo de dar contexto al mismo e indicar, de manera simplificada todos los temas tratados.

- En segundo lugar, se ha desarrollado de manera extensa una sección explicando el estado del arte, se tratan temas relacionados con el proyecto, tales como los vehículos no tripulados, métodos de control de éstos y softwares de programación de los algoritmos de control empleados.
- En el tercer capítulo se encuentra desarrollado el grueso del trabajo, se explica el hardware y software utilizado, así como el control implementado y su programación.
- Posteriormente, en el cuarto capítulo, se analizan los resultados obtenidos con los algoritmos de control implementados para el movimiento autónomo del vehículo.
- Las conclusiones y trabajos futuros se han incluido en dos secciones independientes y posteriores al análisis de resultados.
- Finalmente, el séptimo y último capítulo es la bibliografía referenciada a lo largo de toda la memoria.
- Se ha incluido un anexo A en el que se detallan los procesos de instalación, mediante consola de comandos de Linux de ROS y MAVROS.

2. ESTADO DEL ARTE

En el presente capítulo se va a exponer el estado del arte para las diferentes secciones del trabajo. Se hará una breve explicación de los diferentes tipos de vehículos terrestres no tripulados que existen actualmente en el mercado, se detallarán los algoritmos de control de movimiento autónomo más utilizados en este tipo de vehículos y se comentarán las plataformas disponibles para su programación.

2.1. Vehículo terrestre no tripulado, UGV

Los UGV o vehículos terrestres no tripulados, son un tipo de vehículos, cuyo tamaño, características y aplicaciones varían en un amplio rango según el campo en el que se vaya a implementar su uso, sin embargo, todos ellos tienen en común un aspecto, en ningún caso están tripulados ni pilotados desde su interior [2].

Fig. 2.1. UGV militar [3]

El origen de los UGV data del siglo XX, cuyo objetivo era el control de tanques de manera remota, para así evitar poner en riesgo vidas humanas durante el desarrollo de diversas guerras. Sin embargo, el primer desarrollo de un UGV totalmente funcional fue llevado a cabo por DARPA en la década de 1960 [4]. Éste UGV fue nombrado SHAKEY, y contaba con un sistema de cámaras de televisión, sensores ultrasónicos y táctiles, así como de un sistema de conexión con la estación mediante radiofrecuencia. Todos estos sistemas, junto con una programación que permitía comandos escritos en lenguaje común permitieron que el vehículo llevara a cabo tareas sencillas y repetitivas en entornos controlados.

Uno de los modelos de UGV más conocidos, debido a su popularización en las misiones de la NASA a Marte, es el rover. Es un tipo de vehículo no tripulado, cuyo diseño permite su movimiento sobre la superficie de planetas, lunas o asteroides, con la finalidad de explorar estos, reduciendo al mismo tiempo las posibilidades de sufrir accidentes o averías que puedan interrumpir las misiones espaciales [5].

Fig. 2.2. Rover Mars Curiosity [6]

Los vehículos rover son clasificados atendiendo a diferentes aspectos de estos, tales como su fiabilidad, compacidad, autonomía, o su diseño. En el presente apartado se desarrollará aquella que se ajusta a las necesidades del trabajo, es decir, la clasificación en función del diseño. En ella nos encontramos con tres categorías destacadas, en la que se comprenden los vehículos con direccionamiento, sin direccionamiento, y aquellos que basan su movimiento en sistemas alternativos a las ruedas

2.1.1. Rover con ruedas con direccionamiento.

Uno de los diseños más comunes aplicados a los vehículos rover es aquel que está basado en la estructura y movilidad de los automóviles, es decir, un chasis clásico, al que se le proporciona movilidad mediante cuatro ruedas. Las dos ruedas frontales son las encargadas de la tracción y el direccionamiento del vehículo, mientras que las traseras ofrecen estabilidad y acompañan al movimiento llevado a cabo por las delanteras.

Esta línea de diseño de los rover ha ido variando a lo largo de los años, ya que las condiciones de otros planetas hacen necesaria una mayor robustez en la realización de los

movimientos. Para conseguir una mayor fiabilidad se añade un mayor número de neumáticos, e incluso llegan a incluirse otros cuerpos que añaden estabilidad y movilidad en diferentes planos.

Dos modelos específicos de rover dentro de esta clasificación han estado en la superficie de otros planetas, el rover Lunokhod, cuyo diseño constaba de ocho ruedas y el rover Mars con un diseño de seis ruedas con suspensión tipo *bogie* [7].

Fig. 2.3. Rover con direciconamiento en las cuatro ruedas [8]

Una última variación de los rover con ruedas con direccionamiento es en la que todas las ruedas tienen direccionamiento y tracción. Este modelo específico se conoce como omnidireccional. Pese a ser una variación con muchas posibilidades, su complejidad de diseño y movimiento dificulta su implementación en aplicaciones reales.

2.1.2. Rover con ruedas sin direccionamiento

Con el objetivo de conseguir un movimiento preciso en terrenos especialmente complicados, se lleva a cabo el desarrollo de vehículos rover con ruedas sin direccionamiento. Con este diseño, y gracias a la redundancia en la motricidad, se reducen las posibilidades de desvíos en las trayectorias, e incluso de posibles inmovilizaciones en terrenos especialmente propensos a esto.

Dentro de esta categoría destacan los vehículos con tracción diferencial, que permiten un movimiento más fluido en curvas y giros. Esto es posible, ya que las ruedas de ambos laterales serán capaces de girar a diferentes velocidades, consiguiendo trazar curvas de una manera más precisa que los vehículos de tipo *Skid Steer*, en los que los motores de

ambos laterales siempre giran a la misma velocidad, pudiendo variar únicamente el sentido de giro.

El rover con tracción diferencial es el elegido para el presente trabajo, ya que su diseño facilita la movilidad en los terrenos complicados por los que habrá de moverse. Además, permite realizar giros manteniendo la posición o trazar curvas complejas. Esto es posible debido a la naturaleza de las ruedas, ya que pueden controlarse a igual o diferente velocidad y aceleración, según las necesidades específicas del terreno. Tal y como se explica en posteriores capítulos de la memoria, la tracción diferencial es indispensable para la aplicación desarrollada.

Fig. 2.4. Rover con seis ruedas sin direccionamiento [9]

2.1.3. Rover sin ruedas

También existen diseños de rover que no utilizan ningún tipo de ruedas para moverse. Este es un diseño mucho menos común, pero para determinados terrenos, tales como pantanos u otras acumulaciones de agua poco profundas, e incluso arenas movedizas es una solución factible.

Uno de los diseños más prometedores de este tipo de rover es el llevado a cabo por el departamento de ingeniería de la Universidad de Stanford[2]. El sistema, conforma de cubo, es capaz de moverse en algunas de las situaciones más extremas conocidas [10].

[2] www.engineering.stanford.edu

Fig. 2.5. Rover Hedgehog [11]

2.2. Algoritmos de control para el movimiento autónomo

El movimiento autónomo de todo tipo de vehículos, tanto terrestres como aéreos se ha popularizado y extendido en los últimos meses y años, dando lugar a nuevos métodos de control, sistemas de seguridad y sensores y actuadores más precisos y adaptados al campo de la robótica. Pese a este impulso en el desarrollo de vehículos autónomos, especialmente en aquellos no tripulados, siguen implementándose métodos de control basados en sistemas PID.

2.2.1. Control PID

Los controladores PID basan su funcionamiento en el cálculo de la desviación entre el valor deseado, bien sea este de distancia, velocidad u otros; y el valor real, aquel que es medido por los sensores del sistema. Esta condición hace que aquellos sistemas en los que se desconoce el proceso sean controlables.

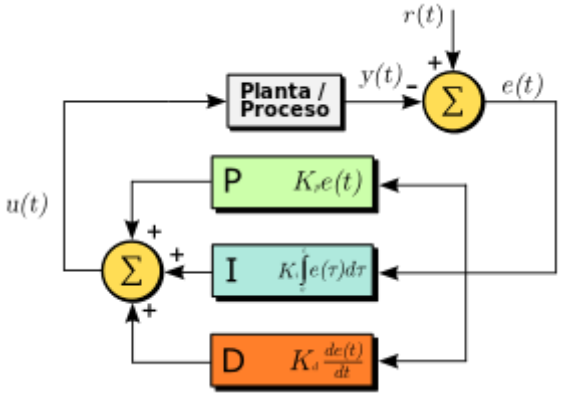

Fig. 2.6. Esquema controlador PID genérico

El controlador PID basa su funcionamiento en tres parámetros, los cuales son, tal y como su nombre indica, proporcional, integral y derivativo. Cada uno de ellos cumple una función específica en el sistema de control. La parte proporcional, busca aproximar a cero el error en estado estacionario, la parte integral tiene un objetivo similar. Ambas difieren en la manera de conseguirlo, ya que el control proporcional consiste en el producto entre el error y la constante proporcional, mientras que el control integral, lleva a cabo una integral entre la variable y el punto de consigna. Por último, la parte derivativa tiene la función de mantener el error al mínimo, mediante la aplicación de derivadas con respecto al tiempo.

En aplicaciones de movimiento autónomo de vehículos terrestres se utilizan tres tipos principales de controladores PID, basados en posición, en velocidad y en una combinación de ambos. Todos ellos realimentados, ya que el tipo de sistema que se maneja requiere conocer el estado previo del sistema, además de por razones de seguridad, para obtener un control más preciso y fiable.

Para desarrollar un controlador PID para aplicaciones de movimiento autónomo, el primer factor que ha de tenerse en cuenta para evitar tener errores evitables en la ejecución de este, es la representación de la posición y la orientación del robot en el plano *xyz*. Además, y con el objetivo de facilitar los cálculos, se llevará a cabo utilizando operaciones elementales. Además, será asimismo necesario ser capaz de representar posibles obstáculos en el mismo sistema de coordenadas, para permitir al robot realizar movimientos que permitan esquivar a estos [12].

Con el objetivo de realizar un control robusto y fiable, es necesario conocer el modelo matemático que representa el sistema, ya que definirá su estrategia de control. El modelado matemático describirá el comportamiento físico del vehículo través de una serie de ecuaciones matemáticas resultado de la definición de una serie de parámetros relativos a su estructura física y sus características técnicas.

Las ecuaciones del sistema además han de apoyarse en lecturas del entorno muy precisas, por ello es necesario utilizar sensores con una elevada precisión, y adecuados para cada aplicación. Ya que, si un determinado sensor, pese a ser preciso, es implementado en una función para la que no ha sido diseñado, el resultado será no satisfactorio. Algunos de los sensores más utilizados para conocer la posición de un determinado robot son transpondedores con sus respectivos lectores, sistemas RFID, ultrasonidos, UWB, Wifi o IR. Cada uno de estos sistemas tiene un campo de aplicación diferente dentro de la

detección de objetos, en función de la lejanía, las posibles interferencias, el tamaño del objeto a detectar o la precisión requerida en la aplicación [13].

Tanto el avance que ha tenido lugar en los últimos años en dispositivos de control, como la mayor precisión que ofrecen los sensores y actuadores implementados, ha permitido a los sistemas de control basados en PID, continuar siendo una de las estrategias de control más utilizadas tanto industrial como académicamente, pese a sus múltiples limitaciones.

2.2.2. Estructura controladores PID

Todos los controladores PID poseen la misma estructura, mostrada en la figura 2.7.

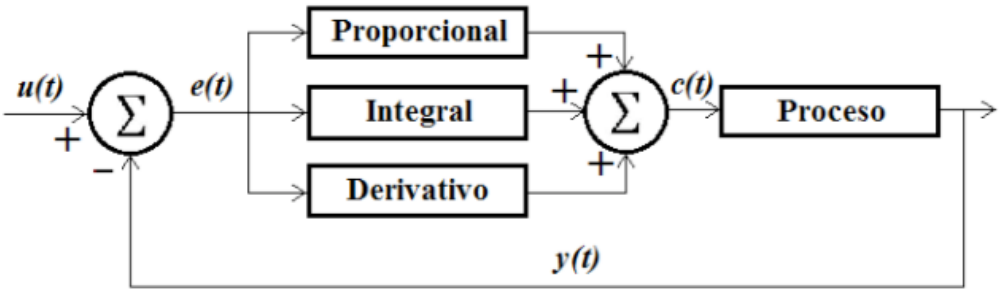

Fig. 2.7. Esquema PID [14]

Los principales parámetros del sistema son:

- u(t): valor de consigna, es decir, el valor deseado.
- e(t): resultado obtenido al comparar, normalmente restar, del valor de salida y(t) con el valor de consigna
- c(t): valor obtenido a la salida del regulador PID

Todos los parámetros mencionados son necesarios para obtener un resultado de salida (real) lo más parecido posible al deseado (ideal). Sin embargo, en función de la señal de entrada y salida del sistema, puede incluirse u omitirse cada uno de los tres parámetros PID, ya que cada uno realiza una función diferente dentro del sistema.

El regulador Proporcional, P, consiste en el producto entre la señal del error y la constante proporcional. Su función es que el error en estado estacionario se aproxime a cero, es decir, reduce el error permanente. Este regulador tiene un valor límite que será el que

minimice la sobreoscilación derivada de su aplicación. La fórmula del regulador proporcional es la siguiente:

$$P = K_p \cdot e(t) \tag{2.1}$$

El regulador Integral, I, actúa cuando existe una desviación entre el punto de consigna y la variable. Tiene la función de disminuir y en algunos casos eliminar el error en estado estacionario. Su funcionamiento, tal y como su nombre indica, consiste en la integración del error del punto de consigna, que posteriormente será multiplicado por la constante K_i, con lo que dicho error se promediará y se sumará. La ecuación que define dicho regulador es:

$$I = K_i \cdot \int_0^t e(\tau) \cdot d\tau \tag{2.2}$$

Por último, el regulador Derivativo, D, tendrá lugar cuando exista un cambio en el valor absoluto del error. Su función consistirá en minimizar el error, corrigiéndolo con la misma velocidad a la que se produce, gracias a su acción derivativa. Su fórmula atiende a la forma:

$$D = K_d \cdot \frac{de(t)}{dt} \tag{2.3}$$

En general, cuando no existe variación en el proceso se implementa la acción proporcional. En los casos que existan errores permanentes, se añadirá el regulador I. Y, en última instancia, cuando existe un cambio en el valor absoluto del error, se incluirá el regulador D, complementando al proporcional.

Combinando la aplicación de cada una de las tres componentes, se da lugar a un total de cuatro posibles controladores, a saber, P, PI, PD y PID. Cada uno de ellos tendrá aplicaciones diferentes en función de los requerimientos del sistema, sin embargo, el método de aplicación será siempre el mismo. Han de seguirse las reglas Heurísticas de ajuste para los reguladores, de manera que la solución obtenida sea la ideal para el sistema que se desea controlar.

2.2.3. Evolución del control basado en PID

Pese a que, tal y como se ha comentado el control PID es uno de los algoritmos de control más utilizados en la actualidad, para minimizar los inconvenientes, se ha desarrollado una nueva filosofía de control que parte de la misma base, pero que sustituye el modelo matemático por ecuaciones diferenciales ultra-locales y fenomenológicas. Dichas ecuaciones relacionarán de manera genérica las entradas con las salidas, en espacios de tiempo reducidos, gracias al factor diferencial de estas. Su funcionamiento básico es el mismo que el PID tradicional, sin embargo, se añade una componente inteligente, cuya función es compensar los efectos de las dinámicas del vehículo no modeladas, por lo que la acumulación de errores disminuirá significativamente, al mismo tiempo que aumenta la precisión [15].

2.2.4. Control óptimo

Una de las alternativas más interesantes desde el punto de vista de la ingeniería a los reguladores PID es el control óptimo. Se trata de una estrategia de control menos extendida, tanto en entornos académicos como en entornos profesionales. Sin embargo, tiene una serie de ventajas con respecto al control basado en PID, que lo hacen más conveniente para ciertas tareas tales como la programación.

El control óptimo destaca en aplicaciones teóricas de diferentes campos de investigación, no necesariamente relacionados con la ingeniería. Una de las áreas de aplicación de la teoría del control óptimo es la economía, sin embargo, dentro del campo de la ingeniería, el control de motores de corriente continua, o el control de dispositivos de flujo son dos de las aplicaciones más comunes, siendo más habituales los planteamientos teóricos que las aplicaciones prácticas de las mismas.

El objetivo del control óptimo es mejorar el funcionamiento de un sistema mediante la aplicación de una función que cumpla las restricciones del sistema haciendo que este funcione de la manera "óptima". Dicha función se denomina *función de coste* o *índice de desempeño*, y permite optimizar sistemas de acuerdo a sus necesidades, ya que el hecho de que un sistema esté optimizado para una determinada aplicación no implica que no pueda mejorarse en un momento dado, únicamente que funciona de la manera óptima en cada caso particular [16].

Esta estrategia de control es la elegida para desarrollar el controlador del rover, ya que además de las ventajas que presenta con respecto a otras aproximaciones tales como los reguladores PID, se trata de un algoritmo con una escasa implementación a controladores para vehículos autónomos.

2.3. Sistemas de piloto automático

En todos los sistemas con movimiento autónomo, bien sean terrestres o aéreos, es necesario un sistema que lleve a cabo el control de nivel bajo, es decir, la conversión de los comandos del software de control en señales eléctricas que se envíen hacia los actuadores.

Esto es posible conseguirlo mediante la programación de un SBC, como puede ser una placa Arduino, la cual permita la conexión de sensores y actuadores tales como motores CC, GPS o encoders entre otros. Esta solución implicaría la programación de dos sistemas diferentes que han de funcionar en conjunto, por un lado, el software de control de nivel alto y por otro el nivel bajo. Sin embargo, existen en el mercado sistemas FCU preensablados, que permiten realizar las funciones de un control de nivel bajo, sin necesidad de programación.

Un sistema de piloto automático es un conjunto de software que permite la comunicación con dispositivos externos tales como GPS, motores u otros actuadores y sensores. Por otro lado, las FCU son microordenadores, en los cuales se instala el software de piloto automático y permiten la conexión de los sensores y actuadores deseados. Sin embargo, ambos términos se utilizan indiferentemente en muchas ocasiones, haciéndose referencia a sistemas FCU con software instalado como pilotos automáticos.

Fig. 2.8. Pixhawk 2 [17]

Actualmente, una de las FCU más conocidos es Pixhawk[3]. Se trata de un módulo FCU, que soporta diferentes softwares de piloto automático, tales como PX4[4] o Ardupilot[5]. Es uno de los sistemas más extendidos, tanto por su alta compatibilidad como por su gran comunidad de usuarios.

2.4. Plataforma de software de desarrollo

Es necesario, para cohesionar y conectar todo el software y hardware del sistema , de manera que todos los elementos del sistema funcionen acorde a las necesidades, una plataforma de software tal como un SO. El sistema operativo tiene que ser capaz de utilizar librerías y otras herramientas, especialmente de comunicación para conseguir un funcionamiento conjunto de todos los elementos de software y hardware. Teniendo en cuenta las necesidades indicadas, se utilizará el sistema operativo de código abierto ROS.

2.4.1. ROS

ROS es un entorno de trabajo flexible creado para desarrollar software para todo tipo de robots. Está formado por una colección de herramientas, librerías y convenciones que facilitan al desarrollador la creación de código [18].

ROS tiene como objetivo proveer diferentes servicios, tales como control de dispositivos de bajo nivel, comunicación entre procesos y nodos, abstracción de hardware y callbacks (frameworks). Además, uno de los mayores beneficios que ofrece es la política de restricciones aplicada al desarrollador, ya que al tratarse de software open source gratuito, no se aplican restricciones de uso de licencia ni de API. Además, y gracias a su amplia comunidad de usuarios en internet, el SO evoluciona y añade mejoras día tras día [19].

[3] https://pixhawk.org
[4] px4.io
[5] ardupilot.org

2.4.1.1. Estructura funcional ROS

El software es capaz de facilitar la inclusión de conexiones entre el software desarrollado de nivel alto, usualmente algún tipo de control; y el nivel bajo del sistema, es decir, el piloto automático al que están conectados todos los sensores y actuadores del sistema.

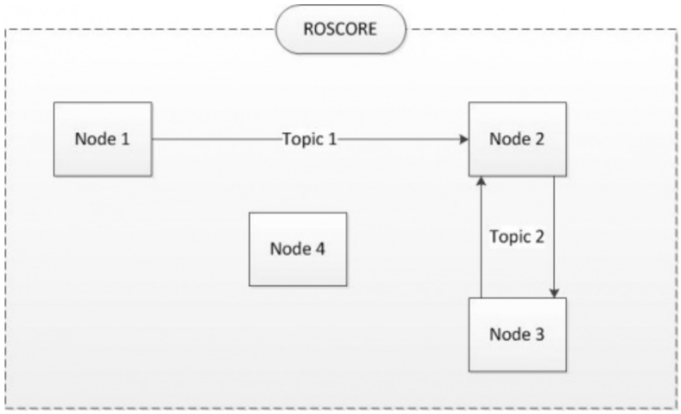

Fig. 2.9. Estructura funcional ROS [20]

La estructura funcional de ROS, independientemente del lenguaje de programación utilizado es invariante. Soporta C++, lenguaje de programación sobre el cual se creó originalmente, además de Phyton y Java, éste último en proceso de implementación. Basa su funcionamiento en una serie de nodos, que pertenecen al mismo ros-pkg y se comunican mediante mensajes pertenecientes al mismo topic o tema. Todo ello ha de estar incluido en el mismo espacio de trabajo dentro del ordenador, sin embargo, es posible la comunicación entre diferentes espacios de trabajo mediante mensajes.

2.4.1.2. Arquitectura ROS

ROS está dividida en tres secciones o niveles de concepto, nivel de sistema de archivos, nivel de computación gráfica y nivel comunitario [21].

El primero de ellos, o nivel de sistema de archivos, consiste en un grupo de conceptos que tratan de explicar la estructura interna de carpetas de ROS y los archivos necesarios para un correcto funcionamiento.

El segundo nivel, el nivel de computación gráfica, es aquel dónde tiene lugar la comunicación entre procesos y sistemas. Aquí ROS maneja los procesos, configura los diferentes sistemas y lleva a cabo la comunicación entre diferentes sistemas.

Por último, el nivel comunitario, ofrece un conjunto de herramientas y conceptos para facilitar la programación a los desarrolladores.

2.4.1.3. Campos de aplicación

Con el objetivo de ser ampliamente utilizado, y abarcar las máximas áreas de desarrollo posibles, ROS fue creado para utilizarse en la mayor variedad de campos de la robótica posibles. Algunos de estos campos son, el movimiento, la percepción, el control, o la planificación. Es especialmente útil en aplicaciones de visión por computador o visión artificial, ya que permite identificar objetos, hacer reconocimiento facial, de gestos y combinación de varias imágenes en tiempo real para obtener una percepción tridimensional del entorno.

2.4.2. MAVROS

Para la comunicación entre el control y el nivel bajo de los diferentes sistemas programados en ROS, es especialmente útil el nodo MAVROS, instalable en el espacio de trabajo de ROS. MAVROS utiliza el protocolo de comunicación MAVLink, esto permite al software de control comunicarse con el sistema de piloto automático.

En función del hardware conectado a sistema FCU, mediante el uso de protocolo MAVLink, pueden obtenerse datos en tiempo real de posición GPS, brújula, aceleración, velocidad, temperatura, presión, batería o imágenes, entre otros. Además, es posible enviar comandos a los actuadores, con mediadas tales como la velocidad objetivo, la aceleración, o una posición deseada. Todo ello facilita la implementación de controladores en sistemas complejos, ya que la comunicación con el nivel bajo se simplifica con el uso del nodo MAVROS.

2.5. Resumen del Capítulo Estado del Arte

En el presente capítulo se expone el estado del arte de los aspectos más importantes que se han desarrollado a lo largo del trabajo. Se ha llevado a cabo un estudio exhaustivo de numerosos trabajos fin de grado y máster, tesis doctorales y artículos de revistas científicas entre otros, centrados en diferentes temas que guardan relación con el desarrollo del controlador para el movimiento autónomo que se ha implementado y probado a lo largo del trabajo.

En primer lugar, se expone el estado actual de los vehículos no tripulados o UGV, atendiendo a su clasificación según la manera en la que llevan a cabo su movimiento, a saber, con direccionamiento, sin direccionamiento y aquellos que utilizan métodos alternativos a las ruedas.

Posteriormente se presenta el estado actual de los controladores para el movimiento autónomo de vehículos, concluyéndose que el método más utilizado es el de los reguladores PID, mientras que el control óptimo únicamente aparece en aplicaciones teóricas que en su mayoría no se aplican al campo de la ingeniería.

Por último, se desarrolla el estado del arte para los sistemas de piloto automático, cuyo mayor exponente en aplicaciones académicas es el sistema Pixhawk, y su integración con el entorno de desarrollo ROS y sus librerías.

3. ALGORITMO PROPUESTO

En la presente sección se presentará el desarrollo llevado a cabo durante el periodo de realización del trabajo. Se explicará además la motivación para la realización de este, los elementos de hardware y software utilizados, así como los algoritmos y teoremas implementados en el control del vehículo.

3.1. Competición buscaminas

El presente trabajo, tiene como objetivo participar en una competición en conjunto con una serie de trabajos realizados de manera paralela por otros alumnos del departamento de Ingeniería de Sistemas y Automática. La mencionada competición, promovida por Minesweepers Towards a Mine-Free World, tiene el objetivo de encontrar y potenciar diferentes soluciones al problema de las minas enterradas en zonas de guerra, para ello cada año desde 2012, proponen una competición en una localidad diferente. Esta competición reúne a equipos en tres diferentes categorías, junior, academia e industria. Cada una de ellas abarca a diferentes perfiles de participantes, menores de entre 10 y 17 años, estudiantes universitarios, y empresas respectivamente.

El objetivo de la competición es detectar una serie de minas, tanto enterradas como superficiales, en un terreno delimitado y dividido en cuadrantes. Para conseguir este objetivo final, la organización redacta una serie de normas que han de cumplirse, así como directrices para el desarrollo del evento [22].

El entorno en el que se realizará la competición es un terreno irregular, cuya superficie tendrá hierba, tierra, zanjas y rejillas. Su superficie total será de 20m x 20m, con divisiones en cada metro cuadrado, de forma que se obtendrá una cuadrícula de 19x19 con medio metro en cada lateral del área de competición como medida de seguridad para los participantes.

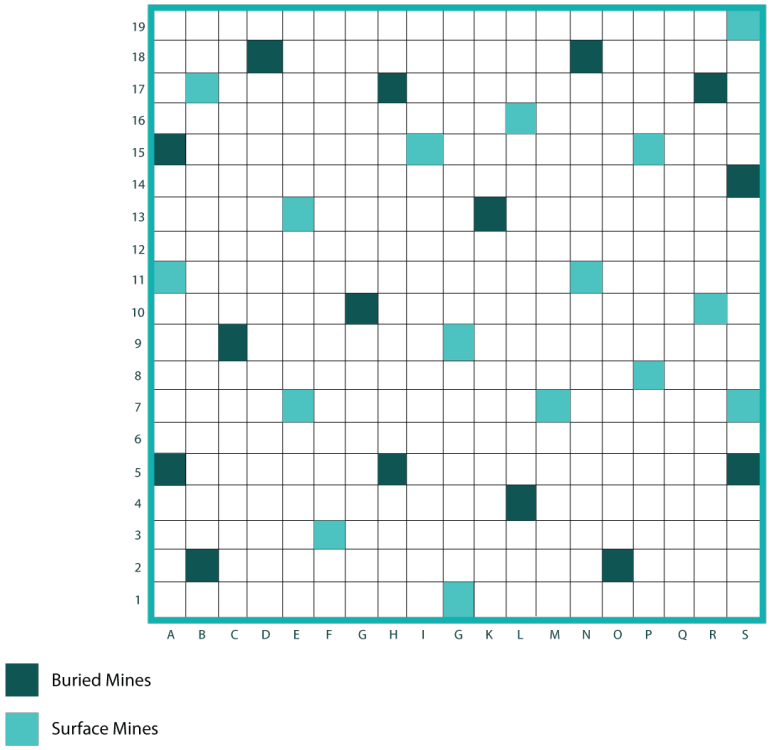

Fig. 3.1. Esquema del terreno, con perspectiva top-down [22]

Las minas pueden encontrarse de dos maneras diferentes, enterradas y superficiales. Pese a esto, el diseño de ambas será el mismo, un cubo metálico, con unas dimensiones de 19cm x 19cm x 19cm. Habrá 16 minas de cada tipo, obteniéndose un total de 32 minas a detectar. Además, tal y como puede observarse en el ejemplo de la figura 3.1, las ocho casillas de alrededor de una mina estarán siempre libres, independientemente del tipo de mina que se encuentre en la misma.

El procedimiento para detectar las minas queda a elección de cada equipo participante. En este caso, se utilizarán dos sistemas diferentes, las minas enterradas serán localizadas mediante un detector de metales, mientras que las superficiales mediante visión por computador. Ninguno de estos procesos se desarrolla en el presente trabajo, sin embargo, guarda una relación directa con los mismos dado que la información que obtengan será la que indicará al controlador del rover la dirección que ha de seguir.

En cuanto a los robots utilizados, quedan a elección de los participantes. Sin embargo, en este apartado existen una serie de limitaciones, tanto en la forma, el tipo de movimiento y el control. En la figura 3.2 se muestra un esquema que resume los tipos de UAV y UGV que se pueden utilizar, en función de su locomoción.

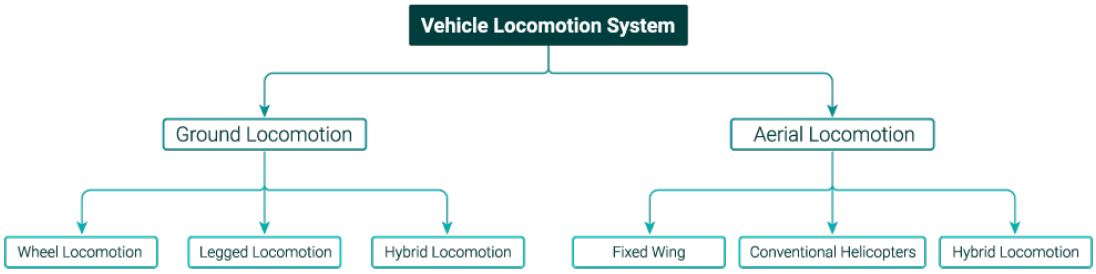

Fig. 3.2. Tipos de robots permitidos en la competición según su locomoción [22]

Los UGV participantes podrán moverse mediante ruedas, patas o un sistema híbrido en combinación de ambas; mientras que los vehículos aéreos, podrán ser del tipo ala fija, helicópteros convencionales, o híbridos. No existen limitaciones en cuanto al método de propulsión, los robots pueden ser eléctricos, neumáticos, alimentados por energía solar o de combustible.

En cuanto al sistema de control, puede ser autónomo o teleoperado, tanto en el caso de los vehículos aéreos como en el de los terrestres. Además, el robot debe haber sido desarrollado por el equipo participante, tanto a nivel hardware como software.

Por último, es necesario referenciar el sistema de puntuación que se aplica en la competición. Cabe destacar la importancia de el uso de vehículos autónomos, multisistema y que implementan ROS, ya que cada uno de estos aspectos por separado supondrá un 20% más de la puntuación total.

El tiempo total de competición para cada equipo será de 20 minutos, incluyendo los tiempos de reinicio en caso de que fueran necesarios. La tabla con todas las puntuaciones parciales y totales se muestra en la figura 3.3, en ella aparecen tanto las acciones que suman puntos como aquellas que penalizan. Las acciones que sumarán puntos serán la detección de minas, tanto enterradas como superficiales; la notificación de la detección y completar correctamente el 80% del terreno. Mientras que, si se falla en la detección, tanto falsos negativos como falsos positivos, se restarán puntos, al igual que los casos en los que las minas detectadas no se notifiquen.

Action	Count	Score / Unit	Subtotal
Arena Score: The following score will be based on the performance of the robot in the competition arena and will be observed and calculated by the in-field judge.			
Detected Surface Mines		5	
Detected Underground Mines		10	
Completely Scan the field and 80% of Mines Detected (Systematic Motion)	Yes No	30 0	
Wrong Detection of a Mine		-5	
Passover Buried Mine without Detection		-10	
Touching Surface Mine		-5	
No light signal and/or a siren	Yes No	-3 0	
Reset Time (-2/1min)		-2	
Mine Map Score: A mine map (in vector or graphical format) must be created automatically by the robot system and presented to the out-field jury committee by the team representative to calculate the following scores:			
Mine Map True Positive for Surface Mines (a minefield cell contaminated by surface mine is labeled in the map as a cell with surface mine)		5	
Mine Map True Positive for Buried Mines (a minefield cell contaminated by buried mine is labeled in the map as a cell with buried mine)		10	
Mine Map False Negative for Surface or Buried Mines (a minefield cell contaminated by surface or buried mine is labeled in the map as a clean cell)		-5	
Mine Map False Positive for Surface or Buried Mines (a clean cell in the minefield is labeled as contaminated by surface or buried mine is the mine map)		-5	
Total Score:			
Autonomous Robot	YES (Multiply by 1.2)		No
Running Using ROS	YES (Multiply by 1.2)		No
Multi Robot System	YES (Multiply by 1.2)		No
Final Score:			

Tabla 4. Plantilla de puntuaciones para la competición [22]

Todas las restricciones y normas mostradas delimitarán el alcance del trabajo y servirán como guía para tomar decisiones en el desarrollo del software y en la elección del hardware del sistema.

3.2. Elección de hardware

Tomando en consideración todas las normas, métodos de puntuación, limitaciones y restricciones impuestos por la dirección del concurso de buscaminas, así como el estado del arte para los diferentes posibles componentes, se procede a la elección del hardware del sistema. En los siguientes epígrafes se desarrollará una explicación sobre cada componente específico, así como una justificación de su elección.

3.2.1. Rover Wild Thumper

Fig. 3.3. Rover Wild Thumper 6WD [1]

El UGV elegido es el Rover Wild Thumper. Incluye únicamente el chasis metálico, preparado para albergar todos los componentes necesarios para su funcionamiento; seis neumáticos de caucho con un diámetro de 120mm y los seis motores correspondientes. Existe además una versión de menor tamaño, con cuatro motores y cuatro ruedas, pero que carece de la estabilidad y la fuerza del modelo elegido.

El diseño del chasis, diferencial sin direccionamiento, tal y como se ha explicado previamente, favorece el movimiento en superficies irregulares y complicadas, lo que permitirá un control de movimiento más preciso.

Es un vehículo ampliamente utilizado, ya que, gracias a su amplio chasis, su moderado precio y motores, es posible implementar una gran variedad de aplicaciones con el mismo. Es posible utilizarlo mediante control remoto o implementando un sistema de navegación

autónomo. Sin embargo, ninguno de estos sistemas se incluye con el rover, de forma que se permite al usuario final elegir el método de control más conveniente para su uso. La única restricción que conlleva el uso de este chasis es el modo de giro, ya que, dado que no posee direccionamiento, será necesario controlar los motores de manera independiente, y realizar los giros en posiciones estáticas, rotando las ruedas de cada lateral en la misma dirección, en sentido opuesto.

3.2.2. Motores DC

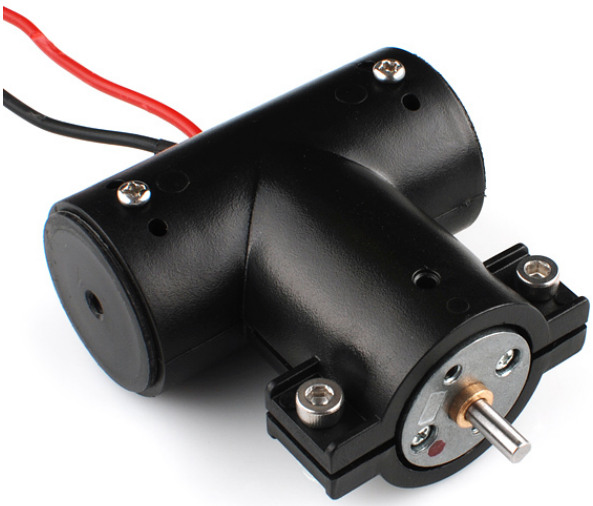

Fig. 3.4. Motor DC Rover Wild Thumper [1]

Cada uno de los seis motores tiene un voltaje nominal de 6.0V_{CC}, con un rango de funcionamiento de entre 2.0 y 7.5V_{CC}. Su corriente sin carga es menor de 350mA y la corriente de rotor bloqueado es inferior a 5.5A. En cuanto al par, con el rotor bloqueado será de 4Kg·cm. Por último, la velocidad de arranque está estimada en 295rpm, con una variación de ±10%, por lo que oscilará entre 325 y 265rpm.

Se trata de motores brushed o con escobillas, por lo que únicamente posee dos terminales para su alimentación, uno positivo y otro negativo, que han de conectarse al terminal positivo y negativo del variador respectivamente.

Dado que hay seis motores independientes, pueden configurarse para trabajar de manera independiente, con instrucciones individuales; o para trabajar como uno solo, conectándose en paralelo al controlador.

3.2.3. ESC HobbyWing QUICKRUN 1060 Brushed

Un ESC o variador es un dispositivo electrónico, que tal y como su nombre indica, tiene la función de variar la velocidad de giro del motor al cual esté conectado. Los ESC se utilizan en gran variedad de aplicaciones en las que se implementan motores, ya sean con o sin escobillas. Las características eléctricas y mecánicas del motor a controlar serán las que determinen que ESC es válido para cada aplicación.

El funcionamiento del variador consiste en la entrega de una potencia eléctrica (voltaje e intensidad) de corriente alterna trifásica en caso de los motores sin escobillas o de corriente continua si el motor tiene escobillas. Los terminales de salida serán tres o dos respectivamente. En cuanto a la entrada al ESC, necesita por un lado alimentación, usualmente entre 5 y 12V, regulada por un módulo BEC. Además, con el objetivo de controlar la potencia eléctrica a la salida, el variador recibe una señal eléctrica, que será directamente proporcional a la entregada al motor.

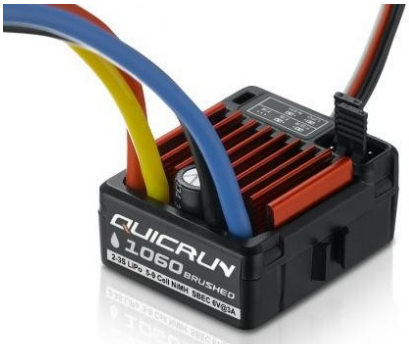

Fig. 3.5. Variador QUICKRUN 1060 Brushed [23]

Partiendo de las características de los motores del vehículo, a saber, 6.0V_{CC} de entrada, y una corriente que ha de oscilar entre 350mA y 5.5A en función de la velocidad de giro y el estado del motor. El ESC elegido es el QUICKRUN 1060 Brushed. Las características técnicas del mismo son:

- Resistencia al polvo y al agua.
- Corriente marcha adelante 60A (Pico de corriente 360A).

- Corriente de marcha atrás 30ª (Pico de corriente 180A).
- Rango de voltaje 2-3S en baterías LiPo o 5-9 en baterías NiMH.
- Tamaño del motor 12RPM o mayor.
- Resistencia desvío 0,001Ω
- BEC salida 2A/5V.
- Frecuencia PWM 1KHz.
- Tamaño 36,5 mm x 32 mm x 18 mm
- Peso de 39g.
- Modos adelante/atrás/freno.

Puede observarse que el ESC elegido será capaz de alimentar, no solo a uno, sino a los tres motores de un lateral del vehículo, ya que cada uno de ellos requerirá un máximo de 5.5A, siendo el variador capaz de otorgar hasta 60A. Este es el principal requerimiento, ya que cada lateral se controlará con un mismo ESC, y la corriente requerida será el triple de la individual de cada motor. Además, y puesto que es necesario moverse en ambos sentidos, además de mantenerse quieto, el ESC elegido permitirá realizar todos los movimientos. Por último, es necesario comprobar la compatibilidad con las celdas de la batería utilizada. Esto se lleva a cabo en la sección dedicada a las baterías necesarias para alimentar al sistema.

3.2.4. Pixhawk 2 (Cube)

Para llevar a cabo el control de nivel bajo del sistema, se ha optado por utilizar un sistema preensamblado de piloto automático, Pixhawk 2 (Cube). Se trata de un dispositivo FCU capaz de ejecutar diferentes softwares de autopiloto, totalmente compatible con ROS y con sistemas UGV.

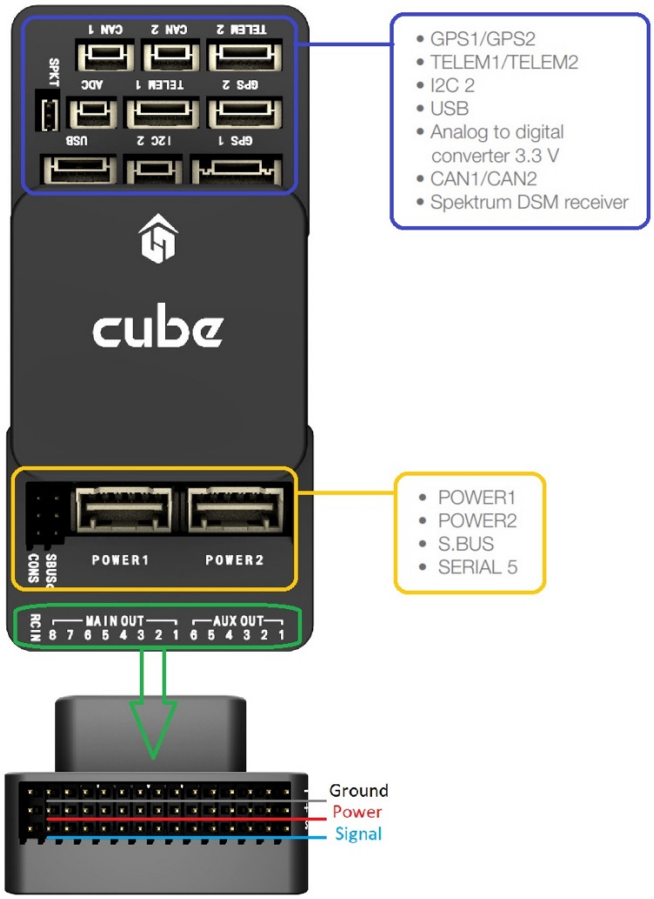

Fig. 3.6. Esquema conexiones Pixhawk 2 [24]

Las características principales del FCU son las que aparecen detalladas a continuación.

- 32bit STM32F427 Cortex M4 core con FPU
- 168 MHz / 252 MIPS
- 256 KB RAM
- 2 MB Flash (totalmente accesible)
- 32 bit STM32F103 coprocesador
- 5x UART (puertos seriales), 2x con control de flujo hardware
- 2x CAN
- Spektrum DSM / DSM2 / DSM-X® entrada compatible con satélite
- Futaba S.BUS®
- Señal de entrada PPM sum
- Entrada RSSI (PWM)
- I2C

- SPI
- 3.3 y 6.6V ADC entradas
- Puerto micro USB interno y extensión micro USB externa

En definitiva, se trata de un sistema potente, ya que, además de procesar los comandos recibidos por el controlador desarrollado en el presente trabajo, el FCU ha de ser capaz de procesar otros comandos de otros sensores utilizados en la competición, así como sistemas de visión por computador. Esto es posible gracias a los 252 millones de instrucciones por segundo que es capaz de procesar la CPU integrada. Además, todos los puertos presentes permitirán la conexión de un máximo de 14 ESC, dos GPS, varios sistemas de telemetría, sistemas conectados mediante puerto serie, y ser controlador mediante USB con conexión directa al ordenador.

Puede concluirse que el sistema Pixhawk 2 ha sido elegido por su potencia, flexibilidad y también por su formato de forma, ya que tiene unas reducidas dimensiones que hacen que sea ideal su implementación en el chasis del vehículo.

3.2.5. Batería

Para la alimentación del sistema, serán necesarias un total de 2 baterías. La primera de ellas será la encargada de suministrar energía a el sistema Pixhawk 2, y los dos ESC, mientras que la segunda provee de electricidad al ordenador Odroid XU4. Ambos sistemas requieren diferentes voltajes de funcionamiento, por lo que la salida de ambas baterías será diferente.

Actualmente existen en el mercado diferentes tipos de baterías, siendo las más comunes en aplicaciones móviles las Li-ion o LiPo. Sin embargo, continúan utilizándose y vendiéndose otros modelos de batería tales como NiCd, NiMH o NiFe.

Las baterías de litio tienen una serie de ventajas con respecto a las fabricadas con níquel, que hacen que sean más versátiles y útiles para aplicaciones móviles:

- La característica más importante es el carecimiento de memoria. Esto implica que las baterías podrán ser cargadas a cualquier capacidad (igual o inferior a la máxima) sin que en próximas cargas esa medida se convierta en la máxima.

- Poseen una tasa alta de descarga, lo cual es imprescindible a la hora de alimentar sistemas que requieren mucha potencia, como el que nos ocupa.
- Tienen una alta densidad, lo que permite que almacenen mucha capacidad en tamaños reducidos y ligeros.
- La tasa de degradación es reducida, llegando a otorgar hasta 1000 ciclos de carga reduciendo su capacidad menos de un 20%.

Este tipo de baterías se utilizan en todo tipo de dispositivos móviles, desde smartphones hasta UAV y UGV, pasando por bicicletas y ordenadores. Sin embargo, su principal desventaja es la necesidad de mantenimiento. Consiste básicamente su calibración cada cierto número de ciclos indicado por el fabricante. Para hacerlo será necesario únicamente descargar por completo la batería y volver a cargarla a su máxima capacidad. Además, es recomendable que, dado que tienden a expandirse y contraerse cuando están cargadas y descargadas respectivamente, que estén almacenadas en contenedores amplios, de forma que se evite cualquier tipo de ignición por contacto de componentes químicos internos.

Es necesario comprobar la compatibilidad de las baterías con los sistemas a alimentar, especialmente los ESC, ya que requieren capacidades y potencias de salida (número de celdas) específicas para un correcto funcionamiento. En concreto, el ESC utilizado, el modelo QUICKRUN 1060, requiere entre 2 y 3 celdas (S) para su correcto funcionamiento. Por esto, y por la corriente necesaria para proveer de electricidad a los diferentes componentes (a excepción del Odroid, que llevará una carga independiente) se ha optado por escoger la batería XXL Power 2600mAh 30C 11,1V.

Fig. 3.7. Batería XXL Power 2600mAh 30C 11,1V.

Las principales características de esta batería, que la hacen óptima para la presente aplicación, son:

- Capacidad típica de 2200mAh.
- Tipo Polímero de litio.
- Voltaje de 11,1V.
- Tres celdas (3S).
- Dimensiones de 105 x 34 x 23.3mm.
- Tasa de descarga típica de 30C (150A).

Como puede observarse en las especificaciones obtenidas del datasheet del producto, será capaz de alimentar a los dos ESC sin problema alguno, ya que posee las tres celdas requeridas para que el motor gire a un mínimo de 12 RPM, en los momentos en los que se encuentre en movimiento.

En cuanto a la alimentación del ordenador a bordo, se ha elegido la batería Desire Power V8 Series 3s 5000mAh 30C.

Fig. 3.8. Desire Power V8 Series 3s 5000mAh 30C

Las principales características de la batería Desire Power son:

- Capacidad típica 1500mAh.
- Voltaje 11,1V.
- Dimensiones de 29.5x45x136mm.
- Peso de 374g.
- Tasa de descarga típica de 30C (150A), y máxima de 60C (300A).

Cabe destacar que, para alimentar el Odroid, pese a que la salida de la batería (11,1V) no coincide con la medida de voltaje necesaria, se utilizará un transformador para ajustar el

mismo a la medida recomendada por el fabricante del microordenador, y así evitar posibles daños irreparables a componentes internos.

3.2.6. GPS Here+ V2 RTK GNSS(M8P)

Un GPS es un sistema que permite localizar un objeto en cualquier punto del globo terráqueo. La precisión de la localización puede variar entre unos metros y pocos centímetros dependiendo del sistema utilizado y de la cantidad de satélites a los que se tenga acceso. Según aumenta el número de satélites, la precisión mejora, sin embargo, en entornos urbanos, debido a la gran cantidad de edificios presentes, es complicado tener acceso a más de 12 o 16 satélites, además, si las condiciones meteorológicas son desfavorables (nubes, lluvia o tormentas) el número de satélites visibles para el GPS se reduce de manera drástica.

Todos los sistemas GPS están compuestos por tres componentes diferenciados:

- Satélites
- Estaciones terrestres
- Receptores

Existe una gran variedad de sistemas de navegación, los más conocidos son GLONASS, desarrollado por la extinta Unión Soviética; Galileo, desarrollado por la Unión Europea; y BeiDou, implementado por la República Popular China. Cada uno de estos sistemas requiere receptores compatibles para su funcionamiento.

El GPS elegido para implementar en el rover, es el modelo Here+ V2 RTK GNSS(M8P), diseñado para funcionar con el sistema Pixhawk 2.

Fig. 3.9. GPS Here+ V2 RTK GNSS [25]

Las principales características de este modelo de GPS son las mostradas en la tabla 5.

Parameter	Specification			
Receiver type	72 channel u-blox M8 engine GPS L1C/A, GLONASS L1OF, BeiDou B1I			
Accuracy of time pulse signal	RMS 99%	30 ns 60 ns		
Frequency of time pulse signal		0.25 Hz…10 MHz (configurable)		
Operational limits [1]	Dynamics	≤ 4 g		
	Altitude	50,000 m		
	Velocity	500 m/s		
Velocity accuracy		0.05m/s		
Dynamic heading accuracy		0.3° [2,3]		
		GPS & GLONASS	**GPS & BeiDou**	**GPS**
Time-To-First-Fix [3]	Cold start	26 s	28 s	29 s
	Hot start	1 s	1 s	1 s
	Aided starts [4]	2 s	3 s	2 s
Sensitivity [5]	Tracking & Navigation [6]	–160 dBm	-160 dBm	–160 dBm
	Reacquisition	–160 dBm	-160 dBm	–160 dBm
	Cold start	–148 dBm	-148 dBm	–148 dBm
	Hot start	–157 dBm	-157 dBm	–157 dBm
Max navigation update rate	RTK	5 Hz	5 Hz	8 Hz
	Moving Baseline RTK	4Hz	4Hz	4Hz
	PVT	5 Hz	5 Hz	10 Hz
	RAW	10 Hz	10 Hz	10 Hz
Convergence time[7]	RTK	<60s[8]	<60s[9]	3.5 min[9]
Horizontal position accuracy	Standalone[9] RTK [6,10]	2.5 m CEP 0.025 m + 1 ppm CEP		

Tabla 5. Datasheet Here+ V2 RTK GNSS [25]

De entre todas las características mostradas, han de destacarse el receptor de 72 canales, dato que permitirá el envío de datos de manera instantánea sin ningún tipo de interferencias con sistemas cercanos. Además, dado que permite el envío de señales con una frecuencia variable, permitirá que el receptor esté situado a distancias lejanas sin dar lugar a pérdidas de conexión o de paquetes de datos.

También, las dimensiones del dispositivo (79x76x16,6mm) hacen que sea ideal su colocación en el UGV, ya que no aumentará significativamente su volumen o peso. Dado que el GPS se ha diseñado específicamente para trabajar con sistemas Pixhawk, su integración es inmediata, y el envío y lectura de datos se produce sin errores. Esto es un factor decisivo en el momento de elegir GPS, ya que de su precisión depende en gran medida el correcto funcionamiento del controlador, tal y como se explica en el capítulo dedicado al desarrollo de este.

Con respecto al sistema RTK, se trata de un dispositivo cuya única funcionalidad es la de mejorar la señal GPS obtenida por el receptor, para obtener una mayor precisión. Esto se consigue conectado el dispositivo RTK a un ordenador mediante un puerto serie, por ejemplo, un USB y situándolo en una posición fija en un lugar elevado, por ejemplo, un trípode. Mediante un software de control terrestre, Mission Planner, se ejecuta un software previamente calibrado que se conectará al dispositivo receptor GPS mejorando su precisión de manera significativa, situándose ésta entre 50 y 20 centímetros, en función del acceso a satélites y el tiempo de calibración del software.

Dado que la aplicación práctica del controlador requerirá una precisión muy alta, este dispositivo, pese a su elevado precio es indispensable, ya que cuanto menor sea el error de medida, las posiciones alcanzadas serán mas similares a las deseadas, puesto que el GPS es el único dispositivo de localización dedicado.

3.2.7. Odroid XU4

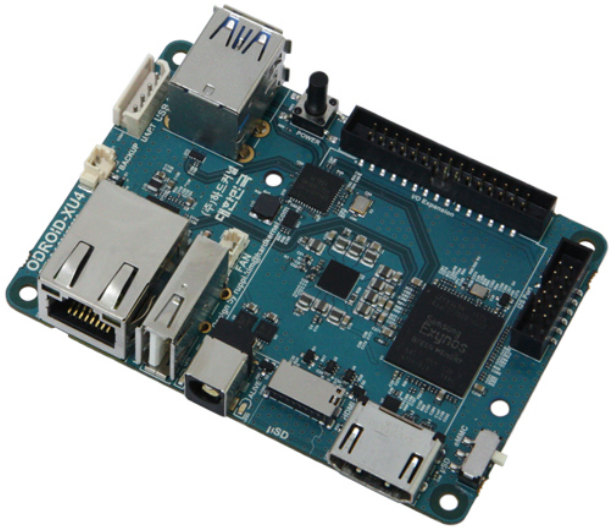

Fig. 3.10. Odroid XU4 [26]

El microordenador Odroid XU4 es un dispositivo de dimensiones reducidas capaz de ejecutar sistemas operativos de escritorio tales como Ubuntu[6] o Android[7]. Soldados en la placa se encuentran todos los componentes necesarios para su funcionamiento, tales como

[6] https://www.ubuntu.com
[7] https://www.android.com

sus procesadores Samsung Exynos5422 y Cortex A7 y su GPU Mali-T628 MP6, además de memoria RAM y almacenamiento.

Este tipo de ordenadores están enfocados a la realización de soluciones de software no comercial. Además, gracias a la potencia de sus componentes internos, especialmente sus dos procesadores, es capaz de procesar las instrucciones de varios nodos ROS, mientras procesa imágenes de las cámaras incorporadas en el sistema.

Sus especificaciones completas se muestran a continuación:

- Samsung Exynos5422 Cortex™-A15 2Ghz y Cortex™-A7 Octa core CPUs
- Mali-T628 MP6(OpenGL ES 3.1/2.0/1.1 y OpenCL 1.2)
- 2Gbyte LPDDR3 RAM PoP
- Almacenamiento eMMC5.0 HS400 Flash
- 2 x USB 3.0, 1 x USB 2.0
- Gigabit Ethernet
- HDMI 1.4a
- Tamaño: 83 x 58 x 20 mm aprox. (excluyendo ventilación)
- Alimentación: 5V/4A
- Linux Kernel 4.14 LTS

Dado que se ejecutarán varios hilos de procesos a la vez en el desarrollo de la competición, y existe un espacio reducido en el chasis del vehículo en el que se incluirán los componentes, este modelo Odroid es una solución ideal, combinando portabilidad y potencia en un sistema que requiere ambas.

3.2.8. RC Turnigy 9X

Pese a que el control utilizado en el rover es autónomo, como medida de seguridad es necesario utilizar un control remoto. El control permitirá activar un modo denominado Offboard, que permitirá al vehículo ser controlado; además es posible armar los servos de manera remota, de forma que en caso de posible accidente o error puede detenerse el movimiento del vehículo de manera instantánea.

Fig. 3.11. Control RC Turnigy 9X [27]

El control elegido para llevar a cabo estas funciones es el modelo RC Turnigy 9X. Cuya característica principal son los 9 canales de comunicación implementados, que trabajan a 2,4Ghz. Esto permitirá controlar tanto el movimiento de cualquier vehículo como otros dispositivos tales como sensores o cámaras que se hayan incluido. Además, posee una pantalla LCD que permitirá modificar ciertas configuraciones, así como observar de manera sencilla parámetros que sea necesario controlar. Por último, cabe destacar su gran rango de funcionamiento gracias a la frecuencia a la que trabaja.

En este caso, no es necesario comprobar la compatibilidad con ningún componente del sistema, ya que este tipo de control es universal, únicamente es necesario que el sistema FCU sea compatible con controladores, algo que se da en la mayoría de los casos, y que tenga un conector para el receptor. Es necesario únicamente elegir las características extra que pueda aportar el controlador al sistema. En este caso son la pantalla y el amplio rango de funcionamiento.

3.3. Conexión del hardware

Una vez se ha elegido el hardware que mejor se adapta a las necesidades del problema propuesto, el siguiente paso, es la conexión de cada uno de los componentes individuales, con el objetivo de que trabajen de manera conjunta sin ningún tipo de incompatibilidades. El esquema de conexión implementado, siguiendo las directrices indicadas por los fabricantes de cada uno de los componentes, se muestra en la figura 3.9. En ella aparece la conexión de cada componente, especificando el o los pines utilizados en cada caso, de manera que se obtenga la mejor configuración posible.

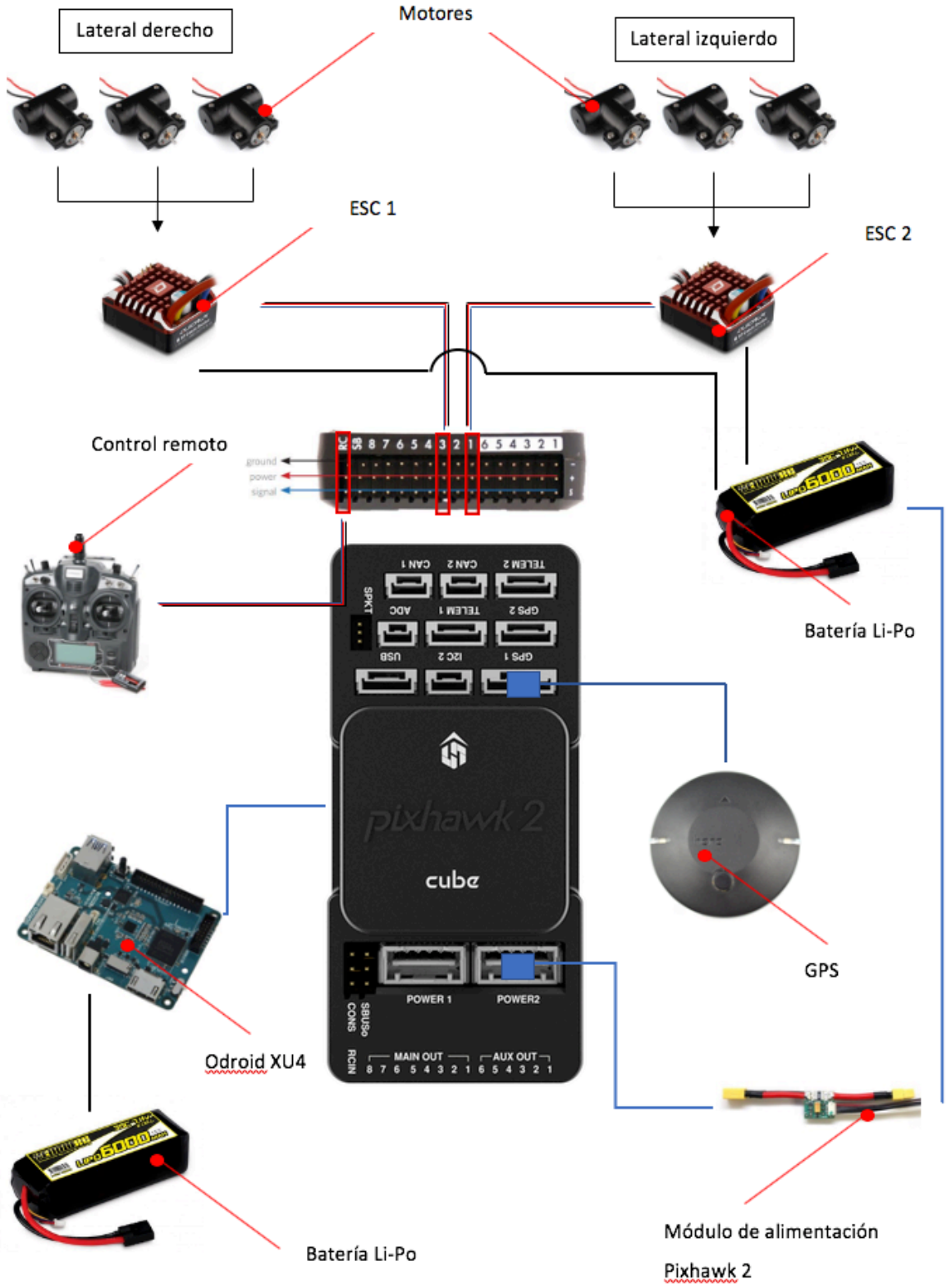

Fig. 3.12. Esquema de conexiones del Rover

En el momento de llevar a cabo el montaje del vehículo, así como sus componentes, es necesario prestar especial atención a tres grupos de conexiones especialmente relevantes, ya que no son necesariamente evidentes y afectan en gran medida al funcionamiento global del sistema si se realizan de manera errónea. Estos son, la conexión de los ESC y la batería con la Pixhawk y la de los motores con los ESC.

El sistema FCU Pixhawk requiere la conexión de los motores de una manera específica en función del método de direccionamiento utilizado. En este caso, dado que el direccionamiento será tipo *Skid Steer*, se conectarán los pines de los ESC derecho e izquierdo a los pines 3 y 1 de las salidas del FCU respectivamente. Esto permitirá un control independiente de cada uno de los sets de motores, para obtener los giros deseados. Cada uno de los 8 sets de pines establecidos como salidas principales, está formado por tres conectores, tierra, alimentación y señal, todos ellos son necesarios para el funcionamiento de los ESC, ya que se alimentan y reciben la señal de control mediante el uso de un conector con tres entradas hembra.

Por otro lado, para alimentar y controlar los seis motores mediante los ESC, se han elegido modelos capaces de proveer de corriente a tres motores. Además, dado que se requiere que los motores de cada lateral del vehículo realicen los mismos movimientos, se conectarán a la salida del ESC en paralelo, de manera que el voltaje sea el mismo en todos ellos y la corriente se divida de manera equitativa y dependiente de la señal de control recibida.

Por último, para alimentar tanto la Pixhawk como ambos ESC se utilizará la misma batería, de manera que se ahorren tanto costes, como espacio y peso en el vehículo. Para ello, la batería se conectará directamente a los ESC con un conector XC60. Mientras que para alimentar a la Pixhawk habrá de incluirse un dispositivo intermedio conocido como módulo de alimentación, que se encargará de suministrar la energía requerida al FCU sin dañarlo. Se trata de un módulo que se incluye con la compra del sistema de piloto automático, y por ello está específicamente diseñado para su alimentación.

El resto de los componentes utilizados se han conectado tal y como se muestra en el esquema. Es especialmente importante conectar de manera correcta el receptor del sistema de control remoto en el sentido debido, ya que es simétrico. Ha de comprobarse que el pin que conectará el cable de señal es el que se encuentra en la parte inferior, y el de tierra en la superior. El dispositivo de localización GPS se acoplará al FCU con el

conector estándar incluido. Y, por último, el ordenador se conectará mediante un cable USB-micro USB estándar a la Pixhawk.

Una vez se han conectado todos los elementos entre sí y se han ubicado y anclado al chasis del rover, el resultado obtenido es mostrado en la figura 3.10. En la misma también se observan otros componentes ajenos al presente trabajo, pero que serán necesarios para la participación en la competición, tales como el detector de metales, que se utilizará para detectar aquellas minas que se encuentren enterradas; o la cámara, cuyo objetivo será, mediante visión por computador, detectar las minas superficiales.

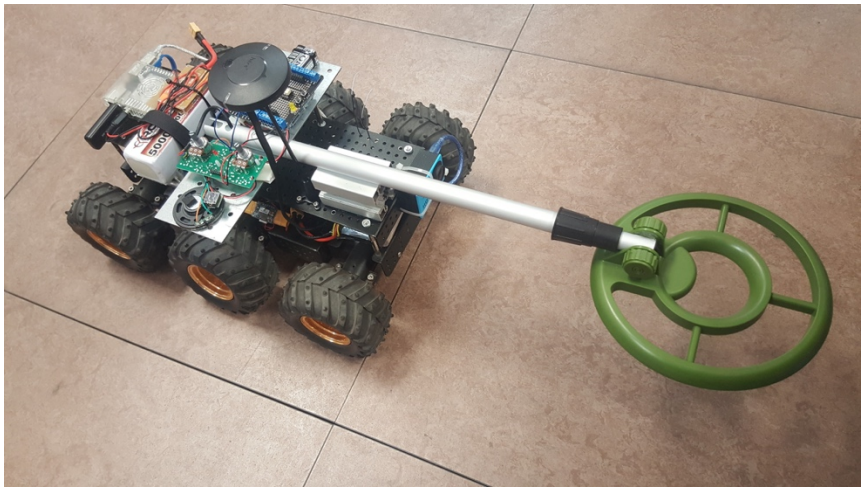

Fig. 3.13. Aspecto definitivo del rover con todos los componentes

3.4. Calibración y pruebas del sistema

Una vez se ha montado y ensamblado el Rover Wild Thumper con cada uno de sus componentes, es necesario calibrar el sistema, así como comprobar el funcionamiento del sistema en conjunto. Para ello es necesario ejecutar un software de piloto automático en el FCU, y utilizar un programa de estación terrestre. Ambos han de ser compatibles entre ellos, así como con la Pixhawk.

El software de autopiloto elegido es ArduPilot, específicamente la versión desarrollada para vehículos rover ArduRover, por su versatilidad, su compatibilidad con Pixhawk 2 y ROS, su orientación a entornos académicos y las opciones de personalización para diferentes tipos de vehículos que presenta. El software PX4 es la alternativa, sin embargo, se ha descartado debido a que su tipo de licencia (BSD, a diferencia de ArduRover que

utiliza GPL) además de su orientación a entornos comerciales, menor compatibilidad con ROS y su implementación específica para rover.

En cuanto a el software de estación terrestre, existen multitud de opciones, con el principal aspecto de decisión relativo a su compatibilidad con el sistema operativo utilizado en el ordenador que ejecuta el programa de control. En este caso, Ubuntu es compatible con todas las opciones disponibles, a saber, Mission Planner, APM Planner, MAVProxy, QGroundControl y UgCS. En este caso, con el aspecto decisivo siendo la compatibilidad entre el piloto automático y el software de estación terrestre, se ha escogido QGroundControl para el desarrollo del proyecto. Este software, además, tiene la ventaja de permitir la instalación de cualquiera de los softwares de autopiloto a través de su interfaz.

El primer paso que realizar en el software QGC será la instalación del software de autopiloto en la Pixhawk 2, ArduRover. Para ello:

1) Se conectará la FCU a un puerto libre del ordenador mediante un cable USB-micro USB.
2) Posteriormente es necesario ejecutar el software de estación terrestre QGC.
3) Se elige la opción firmware dentro de la sección “Vehicle setup” como se muestra en la imagen 3.11. Aquí se escoge la opción ArduRover dentro de el apartado ArduPilot Flight Stack.

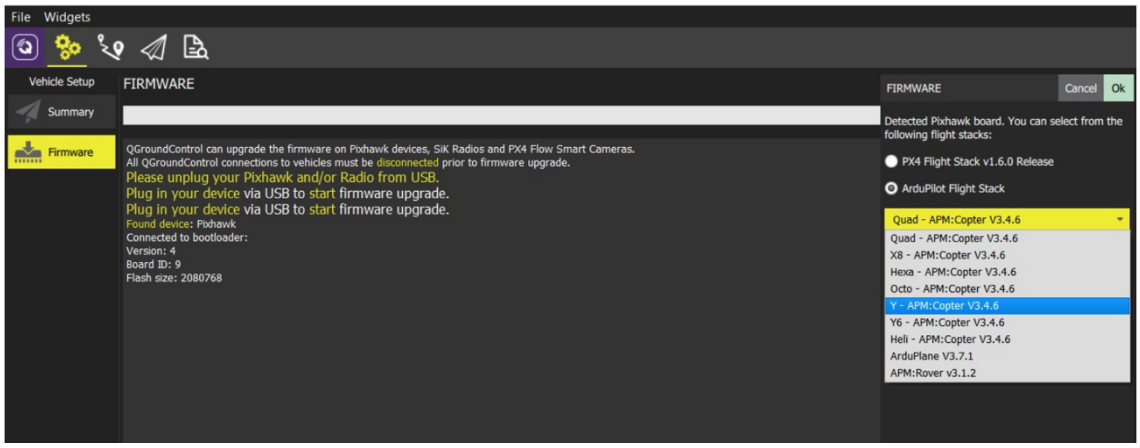

Fig. 3.14. Pantalla instalación software autopiloto

4) Una vez el sistema ha detectado el dispositivo únicamente será necesario pulsar el botón “ok” y esperar varios minutos hasta que se muestre una pantalla similar a la figura 3.12.

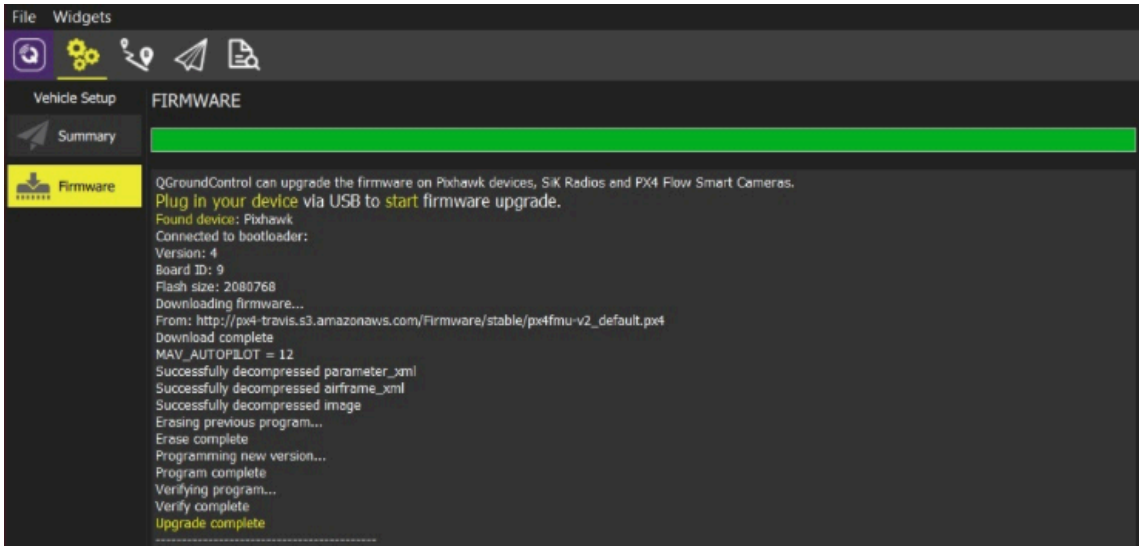

Fig. 3.15. Pantalla con información de instalación correcta

Una vez se ha completado el proceso de instalación del software autopiloto el dispositivo FCU es totalmente funcional. Sin embargo, para una correcta medición de los sensores de este, entre ellos la brújula, el acelerómetro o sensor de nivel, es necesario realizar una calibración para evitar interferencias magnéticas de los componentes. Para hacer estos ajustes, es necesario en primer lugar elegir el tipo de vehículo en el cual se va a implementar el autopiloto, en este caso un rover 6WD.

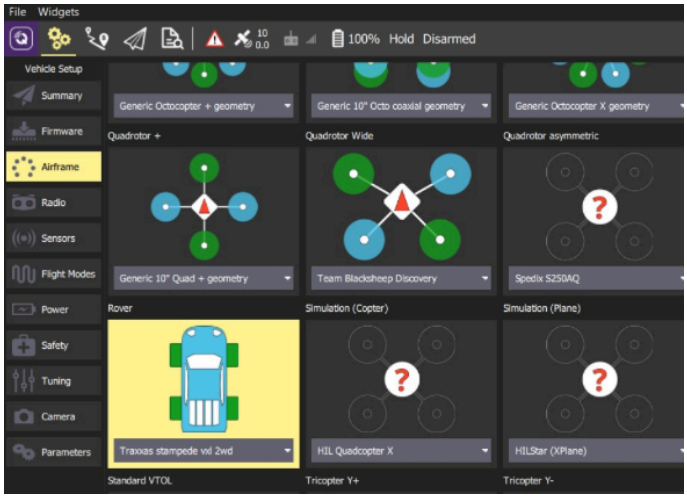

Fig. 3.16. Pantalla QGC de elección de vehículo

Posteriormente se realiza la calibración de los sensores, indicando previamente el sentido en el que se ha orientado la Pixhawk. Las opciones disponibles se muestran en la figura 3.14. La escogida es la primera de ellas, ya que facilita los cálculos de offset a la hora de obtener la orientación del sistema.

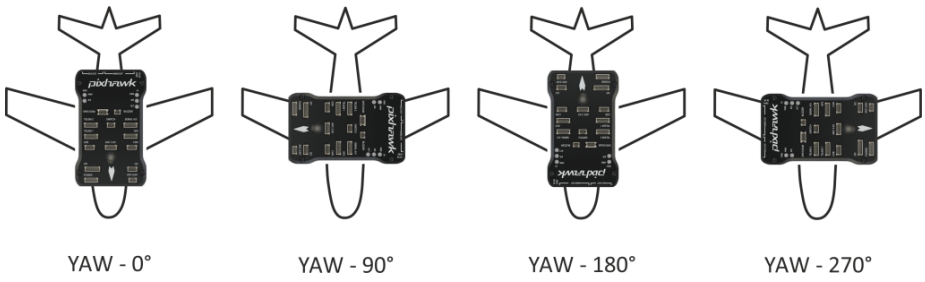

Fig. 3.17. Orientaciones disponibles para Pixhawk [28]

Para realizar la calibración de la brújula, es necesario seleccionar la opción dentro del menú sensores, y seguir los pasos que se muestran en pantalla. Dichos pasos consisten en rotar el vehículo en diferentes ejes hasta que lo indique el software. Esto sirve para ajustar los campos magnéticos generados por el chasis y los componentes del rover, y que afectan a la brújula.

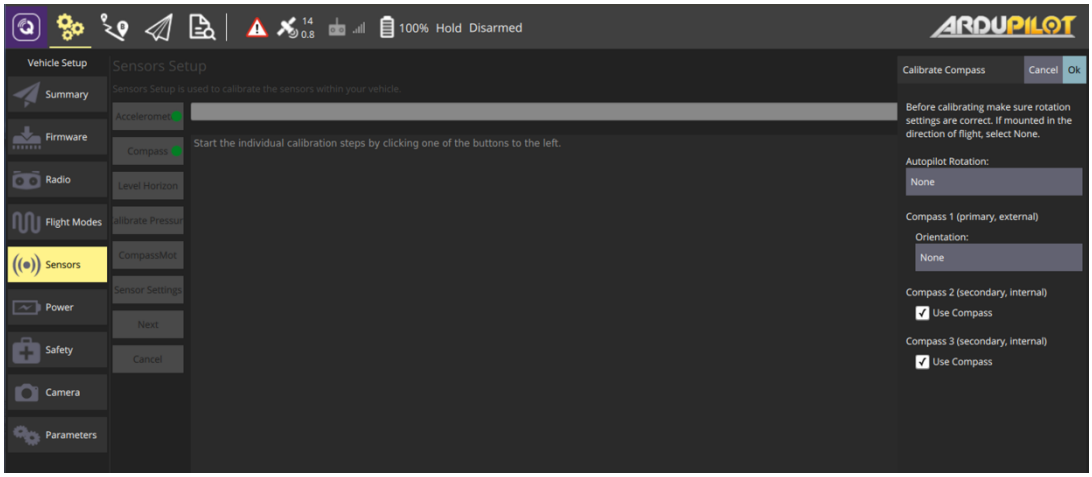

Fig. 3.18. Pantalla de calibración de la brújula

El otro sensor que se ha calibrado, ya que se utilizará en el desarrollo del trabajo, es el acelerómetro incluido en la FCU. En este caso el proceso de calibración es similar, aunque

más sencillo, ya que no será necesario rotar el vehículo, sino únicamente colocarlo en determinadas posiciones, que permitan corregir pequeñas desviaciones en la orientación del dispositivo.

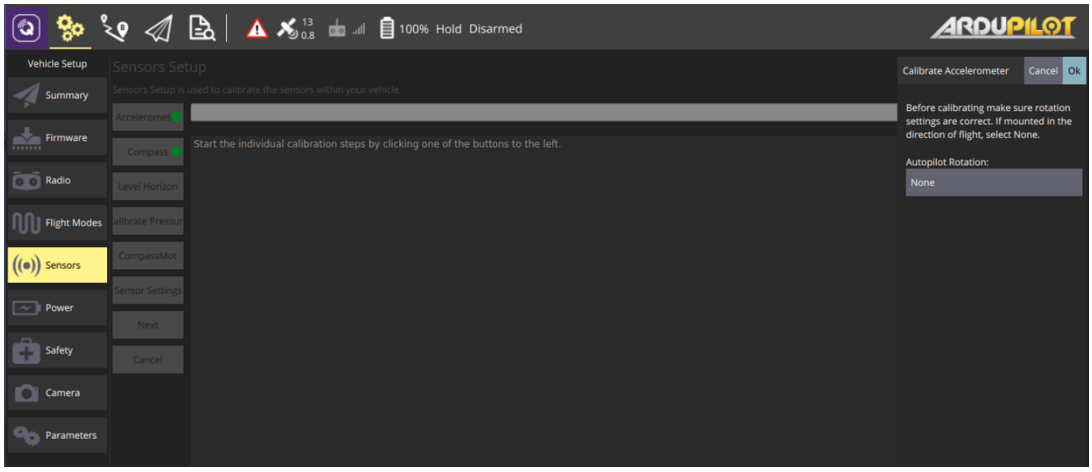

Fig. 3.19. Pantalla de calibración del acelerómetro

En última instancia, para obtener un piloto automático totalmente calibrado y funcional, es necesario conectar el receptor RC a la FCU y configurarlo. De nuevo, esto se lleva a cabo a través del software QGC mediante un epígrafe denominado “Radio” dedicado a esto. Esta calibración consiste únicamente en un mapeo de los botones e interruptores del control remoto. El software muestra una pantalla similar a la figura 3.17 en la que insta al usuario a mover todos los controles para asignar cada uno de ellos a una función diferente.

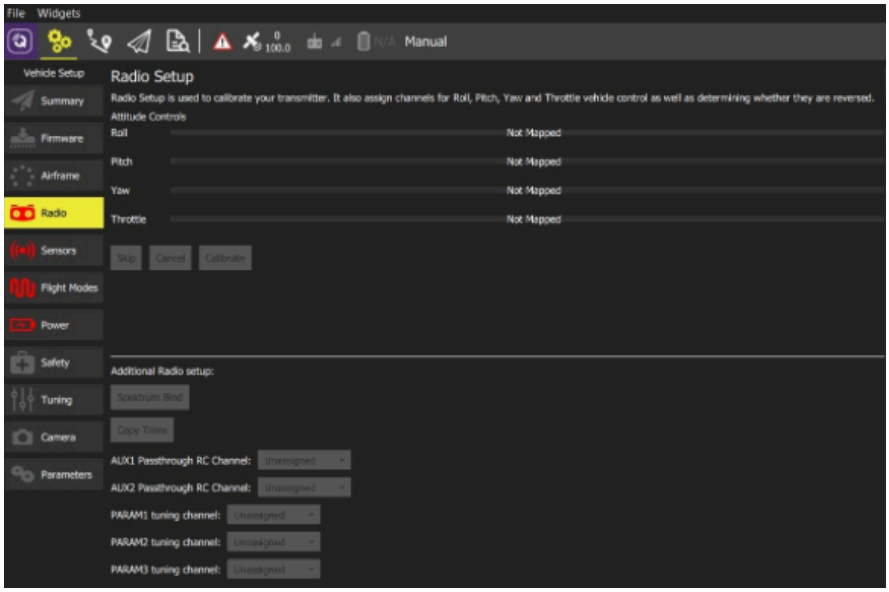

Fig. 3.20. Pantalla de calibración del control remoto

Una vez se han llevado a cabo todos estos pasos, el vehículo se encuentra en un estado totalmente funcional, bien sea para utilizarlo con un control remoto, o mediante un software que controle un movimiento autónomo del mismo.

En nuestro caso, para corroborar que el funcionamiento de todos los componentes es correcto, se han realizado pruebas de movimiento, tanto lineal como curvo utilizando el control remoto. Para ello es necesario activar el modo offboard de la FCU, que permite el movimiento de los motores sin restricciones de seguridad.

Una vez se comprueba que el funcionamiento es adecuado, se continua con el desarrollo del controlador para el movimiento autónomo de rover, tal y como se explica en posteriores epígrafes.

3.5. Diseño de un sistema de control óptimo para el movimiento autónomo de un rover

El sistema de control escogido para el movimiento de rover a lo largo del terreno en el que se desarrollará la competición, está basado en control óptimo, tal y como se ha discutido en la sección del estado del arte. El objetivo primario de este control, siguiendo con las normas de movimiento en la competición, será el de realizar movimientos rectilíneos que cubran la distancia entre los centros de cada una de las celdas contiguas de las que está formado el terreno. Existirán asimismo movimientos de rotación estáticos,

es decir, sin modificar la posición x e y del UGV, únicamente la orientación sobre el eje z. Este movimiento de rotación, con el objetivo de aprovechar las herramientas que se encuentran incluidas en ROS y MAVROS, se llevará a cabo mediante librerías que facilitan la obtención de datos de la brújula de la FCU y la modificación de estos, con movimientos controlados. En la figura 3.21 puede observarse un esquema que representa todos los movimientos posibles que pueden llevarse a cabo por parte del robot, encontrándose el mismo en una celda no exterior del terreno. La única diferencia con las celdas exteriores el es hecho de que el robot nunca se dirigirá hacia los alrededores del terreno.

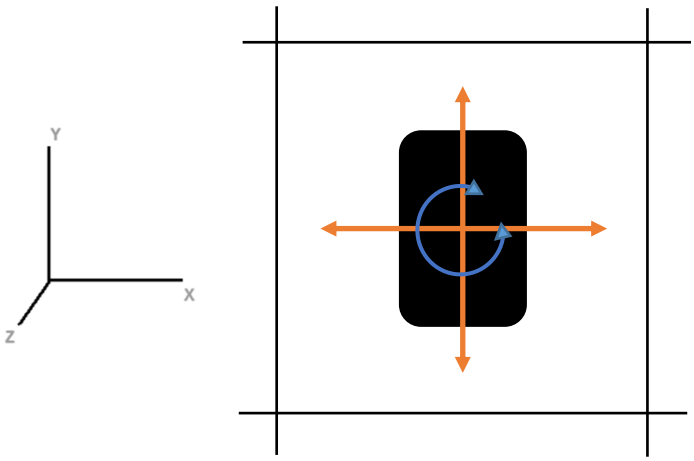

Fig. 3.21. Representación de los posibles movimientos que puede realizar el rover cuando se encuentra en una celda central, desde un plano aéreo

Los dos motivos principales para el uso del control óptimo son, en primer lugar, una mayor facilidad para integrar en el ordenador, es decir, para programar que otras opciones tales como controladores PID. Además, permite especificar ciertas condiciones del movimiento, así como el desarrollo de este, de manera que se adapte a cada aplicación de la manera más precisa posible. Es destacable, asimismo, su menor uso e implementación en sistemas similares, tanto en entornos empresariales como académicos, en comparación con otros métodos tales como el PID y todas sus variaciones.

Para el desarrollo de un algoritmo de control óptimo existen dos métodos destacados, el método de Bellman y el principio del máximo (o mínimo) de Pontryaguin. Ambos métodos tienen en común la definición del sistema a controlar, sin embargo, difieren en la manera de desarrollar el algoritmo. En este caso, debido a las condiciones aplicadas

(tiempo mínimo) que se explicarán posteriormente, se ha escogido el principio del mínimo de Pontryaguin.

El sistema en el que se está trabajando, se define mediante las ecuaciones de movimiento horizontal, definidas en el eje del movimiento, de manera que el sistema obtenido es:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = u \end{cases} \tag{3.1}$$

Dónde x_1 hace referencia a la distancia medida en metros, x_2 a la velocidad, cuyas unidades en el S.I. son m/s y u a la aceleración producto de los seis motores del UGV. Además, ha de tenerse en cuenta que

$$|u(t)| \leq 1 \tag{3.2}$$

Han de indicarse asimismo las condiciones iniciales del sistema, es decir, el punto de partida del movimiento a controlar. Las ecuaciones que definirán el mismo son:

$$x_1(0) = 0 \tag{3.3}$$

$$x_2(0) = 0 \tag{3.4}$$

Por tanto, se parte de un estado de reposo, es decir, sin velocidad, y se considera que la distancia recorrida va aumentando desde cero hasta alcanzar el valor h, de un metro, distancia entre centros de cuadrantes del terreno equidistantes. Por ello, las condiciones finales del sistema serán:

$$x_1(t) = h \tag{3.5}$$

$$x_2(t) = 0 \tag{3.6}$$

Para terminar de delimitar el sistema a controlar, es necesario definir la función de coste más adecuada. Se ha optado por implementar una ecuación de coste que minimice el tiempo de realización del movimiento controlado, ya que, tal y como se ha visto en las normas de la competición, la principal restricción existente en la realización de la prueba es el tiempo habilitado. Por esto, se ha optado por desarrollar el sistema según la siguiente ecuación:

$$J = \int_0^T dt \tag{3.7}$$

Debido a esta elección, el control llevado para el movimiento será un tipo de control óptimo, conocido como *control en tiempo mínimo.*

Para llevar a cabo los cálculos necesarios mediante el principio del mínimo de Pontryaguin, han de seguirse una serie de pasos cuya solución serán las ecuaciones que definirán el movimiento del sistema.

El proceso de optimización del sistema es el siguiente:

- En primer lugar, es necesario determinar el Hamiltoniano del sistema, esto se lleva a cabo atendiendo a la fórmula:

$$H(x, u, \lambda, t) = L(x, u, t) + \lambda^T f(x, u, t) \tag{3.8}$$

  Puede, asimismo, expresarse como función de matrices, lo que facilitará el cálculo de este en el procedimiento posterior. El resultado obtenido será, por supuesto, el mismo independientemente del procedimiento realizado.

$$H(x, u, \lambda, t) = 1 + \lambda^T(t)[Ax(t) + Bu(t)] \tag{3.9}$$

  Además, es necesario tener en cuenta que la ecuación de coste atiende a la fórmula:

$$J = \Phi(x(T), T) + \int_{t0}^{T} L(x, u, t)\, dt \tag{3.10}$$

- El siguiente paso consiste en calcular las ecuaciones de coestado del sistema. Las cuales se obtienen derivando parcialmente el Hamiltoniano con respecto a *x*.

$$\dot{\lambda}^* = -\frac{\delta H(x^*, \lambda, t)}{\delta x} \tag{3.11}$$

- Posteriormente, y partiendo de la ecuación del Hamiltoniano, se calculará la ecuación de estacionariedad.

$$\frac{\delta H(x^*, u^*, \lambda, t)}{\delta u} = 0 \tag{3.12}$$

- Mediante la ecuación de estacionariedad se obtiene la señal de control óptima.

$$u^* = u^*(x^*, \lambda, t) \tag{3.13}$$

- En siguiente lugar, se sustituye en el Hamiltoniano la señal de control óptima que se acaba de calcular.

$$\mathrm{H}^*(\mathrm{x}^*, \lambda, \mathrm{t}) = \mathrm{H}\,(\mathrm{x}^*, \mathrm{u}^*, \lambda, \mathrm{t}) \tag{3.14}$$

- En último lugar, se obtendrán las ecuaciones de estado y coestado, teniendo en cuenta las condiciones iniciales y finales definidas previamente.

$$\dot{x}^* = \frac{\delta H(x^*, \lambda, t)}{\delta \lambda} \tag{3.15}$$

Posteriormente a la definición del proceso a seguir para el cálculo del controlador, se procede a aplicar el mismo al presente sistema a controlar.

En primer lugar, se calcula el Hamiltoniano del sistema, a partir de las ecuaciones que definen el sistema.

$$H\,(x, u, \lambda, t) = 1 + [\lambda_1(t)\ \ \lambda_2(t)]\left\{\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix}u(t)\right\} \tag{3.16}$$

$$H\,(x, u, \lambda, t) = 1 + [\lambda_1(t)\ \ \lambda_2(t)]\begin{bmatrix}x_2(t)\\ u(t)\end{bmatrix} \tag{3.17}$$

$$H\,(x, u, \lambda, t) = 1 + \lambda_1(t)x_2(t) + \lambda_2(t)u(t) \tag{3.18}$$

Una vez obtenido, el próximo paso especificado previamente consiste en derivar parcialmente el Hamiltoniano (3.18), hasta obtener las ecuaciones de estado y coestado del sistema. Esto se realiza de forma que:

$$\dot{\lambda}^* = -\frac{\delta H(x^*, \lambda, t)}{\delta x} = -\begin{bmatrix} 0 \\ \lambda_1(t) \end{bmatrix} \tag{3.19}$$

Obtenida la ecuación general de coestado (3.19), es posible separar en ecuaciones para $\lambda_1(t)\ y\ \lambda_2(t)$, es decir, ecuaciones particulares.

$$\dot{\lambda}^*{}_1(t) = 0 \tag{3.20}$$

$$\dot{\lambda}^*{}_2(t) = -\lambda^*{}_1(t) \tag{3.21}$$

Derivando, se obtienen las ecuaciones de coestado necesarias para proseguir con el procedimiento. Además, dado que los términos evaluados en cero son conocidos pueden sustituirse en la ecuación.

$$\lambda^*{}_1(t) = \lambda_1(0) \tag{3.22}$$

$$\dot{\lambda}^*{}_2(t) = \lambda^*{}_1(t) = \lambda_1(0) \tag{3.23}$$

$$\lambda^*{}_2(t) = -\lambda_1(0) \cdot t + \lambda_2(0) \tag{3.24}$$

En este momento es necesario calcular la ecuación de estacionariedad, a partir del Hamiltoniano calculado en la ecuación 3.18. Será necesario evaluar los posibles valores que tomará dicha ecuación, en función de las ecuaciones de coestado, y con cada uno de ellos proceder a calcular las ecuaciones de estado que definirán el sistema.

$$u^* = \frac{\delta H(x^*, u^*, \lambda, t)}{\delta u} = 0 \tag{3.25}$$

$$\begin{cases} u^* = 1, si\ \lambda^*{}_2(t) < 0 \\ u^* = -1, si\ \lambda^*{}_2(t) > 0 \end{cases} \tag{3.26}$$

Una vez se ha evaluado la ecuación de estacionariedad, y se han obtenido los valores que tomará, se procede a mostrar dicha ecuación como una función dependiente del coestado, para ello se utilizará la función signo.

$$u^* = -sign\left(\lambda^*{}_2(t)\right) = -sign\left(-\lambda_1(0) \cdot t + \lambda_2(0)\right) \tag{3.27}$$

La forma de la ecuación obtenida para la estacionariedad puede representarse gráficamente tal y como se muestra en la figura 3.21.

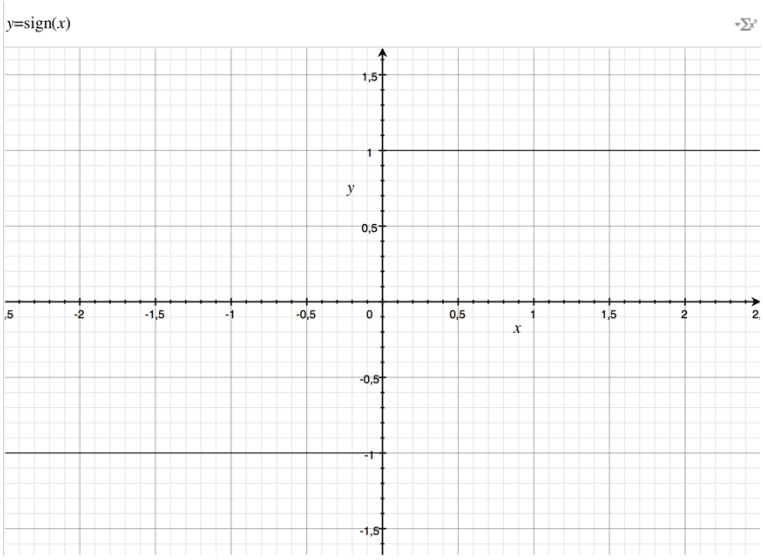

Fig. 3.22. Representación gráfica de la ecuación signo

Una vez se han definido los valores que tomará u^*, se procederá a realizar el cálculo de las ecuaciones de estado del sistema para cada uno de los dos.

Para $u^* = 1$, y con las ecuaciones que definen nuestro sistema, se obtiene lo siguiente:

$$\dot{x}_1(t) = x_2(t) \tag{3.28}$$

$$\dot{x}_2(t) = 1 \tag{3.29}$$

$$x_2(t) = t + x_2(0) \Rightarrow \mathrm{t} = x_2(t) - x_2(0) \tag{3.30}$$

$$\dot{x}_1(t) = x_2(t) = t + x_2(0) \tag{3.31}$$

$$x_1(t) = x_2(0) \cdot t + \frac{t^2}{2} + x_1(0) \tag{3.32}$$

Sustituyendo el valor calculado para t, se obtiene:

$$x_1(t) = x_1(0) + \frac{[x_2(t) - x_2(0)]^2}{2} + x_2(0) \cdot [x_2(t) - x_2(0)] \tag{3.33}$$

Simplificando y resolviendo cuadrados se obtiene la primera ecuación que definirá el movimiento a realizar por el rover, durante el proceso de aceleración que está definido por la restricción $u^* = 1$.

$$x_1(t) = \frac{x_2(t)^2}{2} + \left(x_1(0) - \frac{x_2(0)^2}{2}\right) \tag{3.34}$$

Si despejamos el valor desconocido, es decir, la velocidad en tiempo t, obtenemos la ecuación en su forma requerida para la programación.

$$\boxed{x_2(t) = \sqrt{2(x_1(t) - x_1(0) + \frac{x_2(0)^2}{2})}} \tag{3.35}$$

En el caso de definido por $u^* = -1$, el procedimiento a seguir es idéntico al mostrado entre las ecuaciones (3.28) y (3.34). En primer lugar, se sustituye el valor de u^* en $\dot{x}_2(t)$.

$$\dot{x}_2(t) = -1 \tag{3.36}$$

$$x_2(t) = -t + x_2(0) \Rightarrow \mathrm{t} = -x_2(t) + x_2(0) \tag{3.37}$$

$$\dot{x}_1(t) = x_2(t) = x_2(0) - t \tag{3.38}$$

$$x_1(t) = x_2(0) \cdot t - \frac{t^2}{2} + x_1(0) \tag{3.39}$$

De nuevo, se sustituye el valor de t calculado en la ecuación (3.36), en la ecuación (3.38), obteniéndose:

$$x_1(t) = x_1(0) - \frac{[-x_2(t) + x_2(0)]^2}{2} + x_2(0) \cdot [-x_2(t) + x_2(0)] \tag{3.40}$$

Simplificando se obtiene la ecuación (3.40) que definirá el movimiento del rover durante el proceso de deceleración o frenado.

$$x_1(t) = -\frac{x_2(t)^2}{2} + \left(x_1(0) + \frac{x_2(0)^2}{2}\right) \tag{3.41}$$

De nuevo, se despeja el valor deseado, $x_2(t)$, y se obtiene la siguiente ecuación.

$$\boxed{x_2(t) = \sqrt{2(\,x_1(0) + \frac{x_2(0)^2}{2} - x_1(t))}} \tag{3.42}$$

Ambas ecuaciones obtenidas, una vez programadas, en lenguaje C++, comandarán el movimiento rectilíneo que ha de realizar el UGV. El movimiento se llevará a cabo en función de la posición y velocidad previas del rover, denotadas por (0), y la posición actual, lo que permitirá calcular la velocidad ($x_2(t)$) a la que debe moverse el rover en ese preciso instante de tiempo. Con la realimentación de la que provee el estado previo, pueden calcularse las condiciones de velocidad próximas que a las que circulará el rover, en primer lugar, atendiendo a la ecuación de aceleración, y una vez se ha sobrepasado el punto intermedio del recorrido, es decir, 0,5 metros, las de deceleración.

En la figura 3.22 se han representado ambas ecuaciones gráficamente, atendiendo a diferentes valores, que se definirán de manera específica para el rover en función de sus capacidades mecánicas, en conjunción con todas las restricciones definidas por las necesidades externas del problema, como son posición y velocidad iniciales y finales.

La gráfica izquierda hace referencia a la ecuación para la aceleración, mientras que la derecha es para la deceleración.

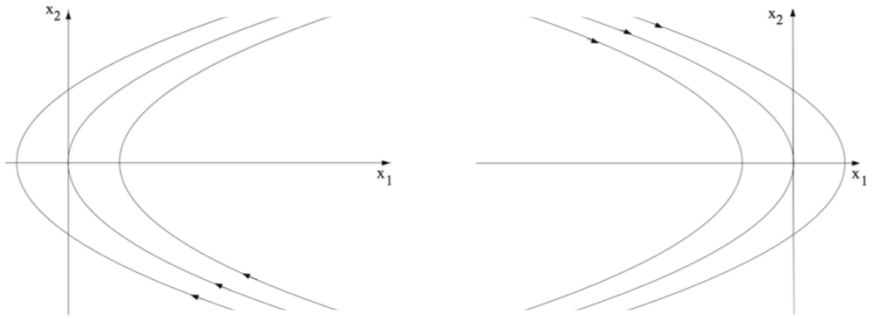

Fig. 3.23. Representación gráfica de las ecuaciones que definen el movimiento del rover. A la izquierda se muestran las que tendrán lugar durante la aceleración, mientras que las de la derecha hacen referencia al proceso de deceleración

Ambas rectas habrán de cortarse en un punto, el cual definirá el momento en el cual el rover ha de terminar de acelerar, para comenzar a decelerar. Dicha intersección, puede calcularse resolviendo el sistema de ecuaciones formada por (3.34) y (3.40). Resolviendo este sistema, se obtiene el punto de cambio x_1.

$$x_1(t) = \frac{h}{2} \tag{3.43}$$

Tal y como se esperaba, el punto de cambio es el punto intermedio del recorrido, esto implica que, para recorrer la distancia de la manera más rápida posible (condición *J*), deberá acelerarse al máximo (nivel definido por el motor) hasta el punto intermedio del recorrido, y en dicho punto comenzar a decelerar hasta alcanzar el punto de destino, en este caso 1 metro, obteniéndose una posición final de reposo.

3.6. Programación del sistema de control

Una vez se han obtenido las ecuaciones que definen el movimiento rectilíneo de traslación entre los centros de dos celdas, es necesario convertir las mismas en lenguaje entendible para el ordenador que se encargará de comandar a todo el sistema. En el presente epígrafe, se mostrarán los diagramas de flujo que muestran el comportamiento del software desarrollado en lenguaje C++ que permite ejecutar el algoritmo de control óptimo implementado para el movimiento autónomo del UGV. También se explicarán aquellos nodos de MAVROS que se utilizan para el envío y recepción de información entre el ordenador ejecutando el programa de control y el sistema de piloto automático.

El funcionamiento general del programa que comanda los movimientos rectilíneos sigue el esquema mostrado en la figura 3.24. En ella puede observarse el diagrama de flujo simplificado de movimiento, ambos puntos de decisión referentes a la distancia recorrida derivan en dos procesos inversos, de aceleración y deceleración. El primero de ellos tiene lugar en los instantes en los cuales se ha recorrido menos de la mitad de la distancia total, y el posterior ser lleva a cabo una vez se ha superado el punto intermedio del recorrido.

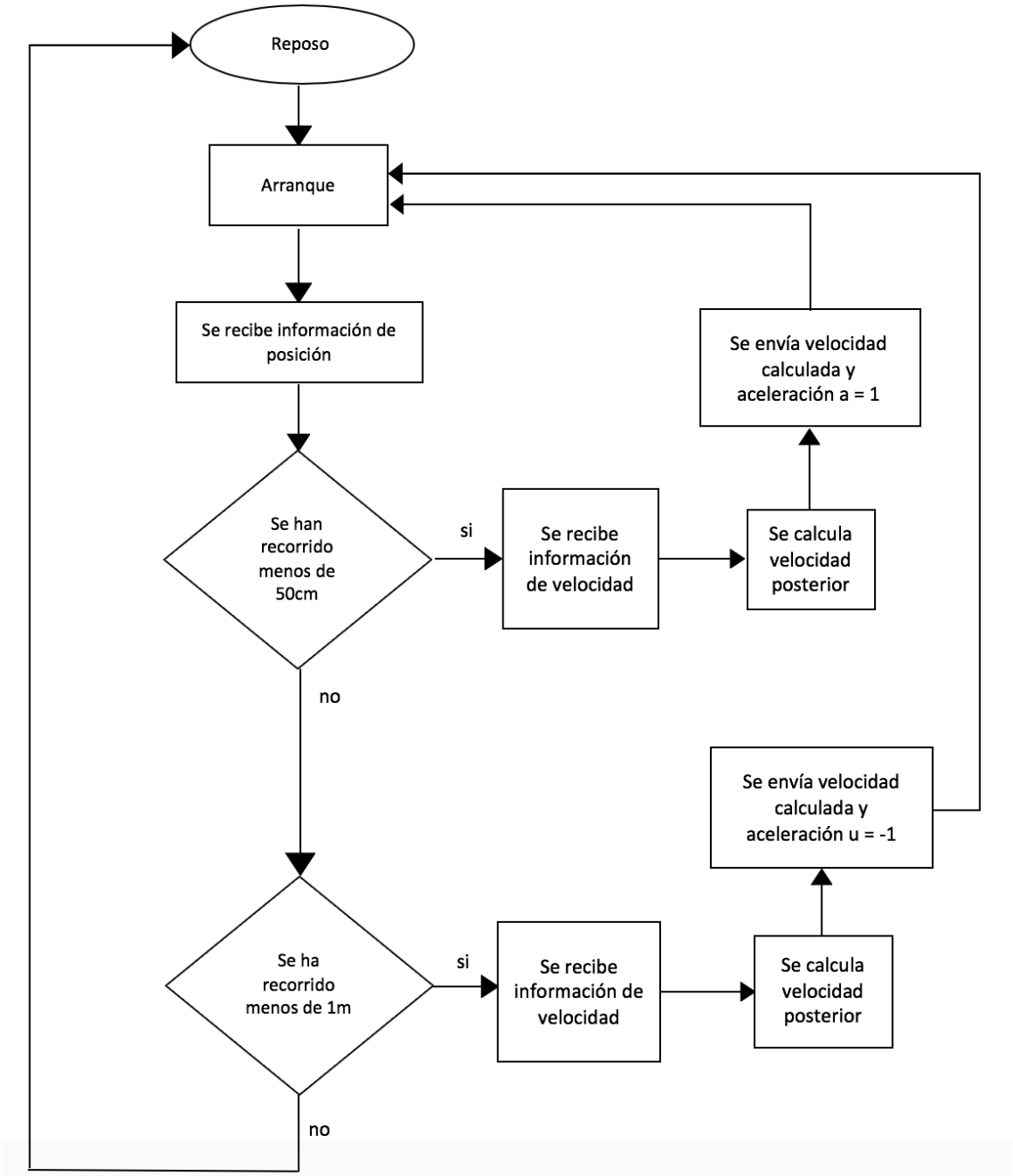

Fig. 3.24. Diagrama de flujo que representa el proceso de movimiento rectilíneo seguido por el rover

En el diagrama puede observase que el UGV estará en un estado de reposo del que saldrá en el momento en el que se active el modo offboard del sistema. En ese momento se activa el vehículo, estado representado por el termino "Arranque", haciendo referencia al estado de los vehículos una vez se encienden sin haber comenzado la marcha. Posterior a esto, se encuentra la primera toma de decisión. La primera opción es la de haber recorrido una distancia menor de 0,5 metros, es decir, menos de la mitad del recorrido que se llevará a cabo en cada tramo. En caso afirmativo se procederá a ejecutar el proceso de aceleración.

El proceso de aceleración tiene la función de calcular la velocidad a la que ha de moverse el UGV en la posterior iteración, mediante las ecuaciones desarrolladas basadas en control óptimo. Posteriormente ha de enviar dicho comando de velocidad, junto con el de aceleración referente al tramo, que se ha definido en el momento de calcular dichas ecuaciones, u=1 (en valor relativo). El cálculo se lleva a cabo conociendo las posiciones y velocidades previas y la posición en la que se ejecutará la velocidad que se está calculando.

Las posiciones se estiman mediante dos procesos. El primero de ellos consiste en la recepción de datos por parte del GPS, como una distancia absoluta medida en variables *x*, *y* y *z*, con respecto a un punto de origen generado por la FCU. De esta manera, y aplicando el teorema de Pitágoras, puede calcularse la distancia recorrida entre dos localizaciones indicadas por el GPS. Cabe destacar, que pese a la presencia de la variable z entre los datos recibidos, únicamente se tienen en cuenta *x* e *y*. Esto es así, ya que el cálculo de z se lleva a cabo mediante un barómetro incorporado en la FCU y dado que el rover se moverá en el plano, no es un dato relevante para la ejecución. El segundo método utilizado es la odometría, consiste en la estimación de la posición atendiendo a la velocidad alcanzada durante un determinado periodo de tiempo. La fórmula utilizada para el cálculo de la distancia por odometría es la siguiente.

$$x[m] = v\left[\frac{m}{s}\right] \cdot t[s] \tag{3.44}$$

De esta manera se obtiene una aproximación de la distancia recorrida, sin embargo, en situaciones reales en las que la velocidad no ha sido idéntica durante todo el periodo, o en aquellos escenarios donde la superficie no es perfectamente plana, la estimación carece de precisión. Por ello, en cada estimación de posición que se lleve a cabo durante la ejecución del controlador, únicamente se tendrá en cuenta la medida de distancia obtenida

mediante odometría en los casos en los que la diferencia entre ambas sea superior al 20%. En estos casos, se realizará una media de ambas medidas, reduciendo de esta manera el error de cálculo.

Siguiendo con el flujograma, una vez se ha sobrepasado la barrera de los 50 centímetros, en la primera decisión, se optará por la opción no, de manera que se escoja la posibilidad en la que se ha recorrido menos de 1 metro. Esto llevará al rover a realizar el proceso de frenado.

En el caso de la deceleración, el procedimiento es el mismo, obteniéndose la información de posición mediante el GPS y la odometría, y de velocidad mediante el acelerómetro de la FCU. Se llevan a cabo la misma serie de cálculos hasta obtener la velocidad deseada, y se envía a los ESC el comando de la velocidad calculada, y aceleración u = -1.

En última instancia, en el momento en el que se supere la distancia objetivo de 1 metro, se procederá a indicar a la FCU que la velocidad deseada es de 0m/s, por lo que el vehículo se detiene y vuelve al estado de *Reposo*.

En cuanto al movimiento de giro, como ya se ha explicado, se implementará haciendo uso de las librerías de ROS. Será necesario obtener la información de la brújula incorporada en el sistema de piloto automático y enviar un comando de velocidad angular al mismo, para que las ruedas reproduzcan un movimiento de rotación. MAVROS regula el movimiento de las ruedas haciendo uso de un regulador PID interno, cuyo funcionamiento consiste en regular la velocidad comandada a las mismas, y ajustarla en función del feedback o realimentación recibida por los componentes de la Pixhawk, tales como la brújula o el GPS. En los casos en los que se comande una velocidad y el vehículo se mantenga estático, dicha velocidad aumentará hasta que se logre superar el obstáculo que impide el movimiento. Una vez se alcance el ángulo deseado con dicha posición, el vehículo parará y se mantendrá a la espera de la siguiente instrucción. En este caso no se ha incluido un flujograma ya que la ejecución es invisible al usuario final, a diferencia de lo que ocurre en el caso del movimiento rectilíneo.

3.6.1. Comunicación con Sistema Pixhawk

Para poder llevar a cabo la recepción y el envío de datos de y hacia la FCU, es necesario utilizar una librería de ROS denominada MAVROS. Mediante una serie de comandos disponibles en esta, es posible obtener la velocidad a la que se está moviendo el vehículo,

así como su orientación y además enviar comandos de aceleración lineal, y velocidad lineal y angular, entre otros muchos. Además, se tendrá constancia del estado de la batería y su vida útil, para recargarla o reemplazarla en los momentos en los que sea necesario.

Los mencionados topics de MAVROS son:

- *Global Position*, específicamente los datos de la sección Local, que permitirá al controlador obtener la posición, con variables de *x*, *y* y *z*, en función de un sistema de ejes de referencia fijos calculados por la FCU.
- *Compass_hdg*, permitirá al controlador encargado del giro conocer el ángulo en grados en el que se encuentra el rover.
- *TwistStamped*, tiene la función de enviar comandos de velocidad lineal y angular, así como obtener los datos de la velocidad actual del vehículo.
- *TF*, permite obtener datos de orientación facilitados por la brújula de la FCU, así como el GPS, cuyo sistema referencia está basado en cuaternios, por lo que requiere conversión a grados o radianes.
- *Thrust*, lleva a cabo el envío de datos de aceleración lineal al rover.
- *BatteryStatus,* tiene la función de reportar datos de la batería en tiempo real, tales como su vida útil, velocidad de descarga y carga restante.

La programación de los mencionados nodos dentro del controlador en ROS permite que se realicen todos los movimientos necesarios de manera satisfactoria.

3.7. Integración del sistema

Para el funcionamiento completo del sistema, es decir, el controlador en conjunto con otros nodos, encargados de diferentes tareas tales como la detección de minas enterradas, la detección de minas superficiales, o la planificación de ruta, es necesario implementar nodos “Publisher” y “Subscriber” de ROS. Estos nodos son los encargados de enviar y recibir la información necesaria en cada instante de la ejecución del programa. El controlador desarrollado en el presente trabajo tiene la necesidad de recibir información de la ruta, es decir, la dirección en la que ha de moverse, y ha de informar del momento en el que ha alcanzado el destino definido. Esto se lleva a cabo según el diagrama de flujo mostrado en la figura 3.25.

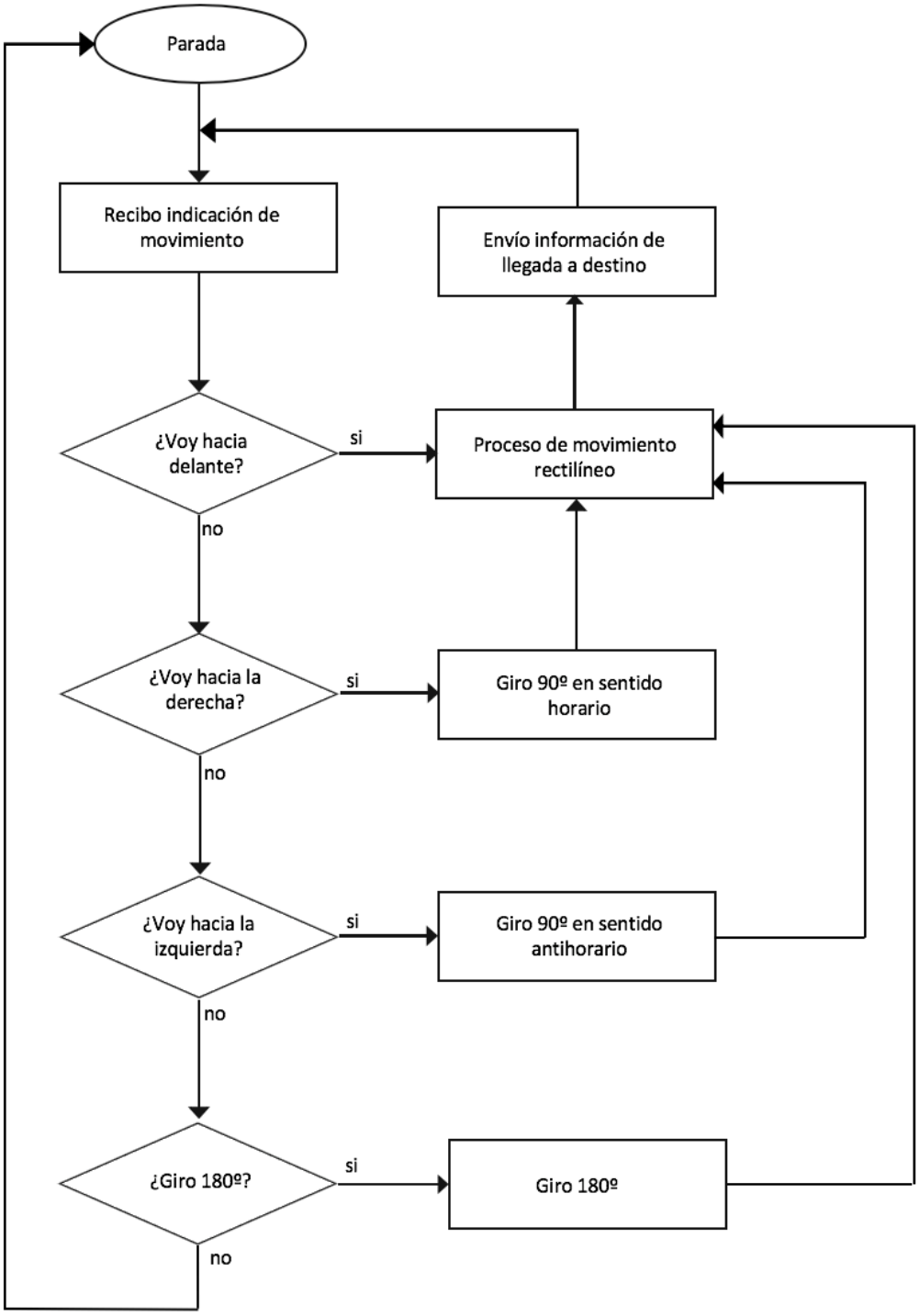

Fig. 3.25. Diagrama de flujo que representa el proceso de movimiento del UGV con la información externa recibida.

En el flujograma puede observarse que, se comienza en un estado de parada. En primer lugar, se recibirá información relativa a el sentido y la dirección de giro hacia la que ha de dirigirse el UGV. Puesto que hay cinco opciones posibles, se colocan de manera que aquellas más probables se comprueban primero. En el caso de que se solicite un

movimiento hacia delante, se ejecutará el proceso de movimiento rectilíneo representado en la figura 3.24 y una vez se complete, se informará de ello al sistema de planificación de ruta. En caso de realizar un movimiento que requiera una rotación, se ejecutará el proceso de giro y posteriormente se moverá al rover de manera rectilínea hacia el destino, de igual forma que en el primer caso, y por último se informará. Los movimientos de giro de 90º se harán en los casos en los que se encuentre una mina en la celda contigua, mientras que el de 180º se llevará a cabo cuando el UGV se encuentre en alguna fila o columna exterior, y haya minas en la celda lateral hacia la cual se está dirigiendo.

El último de los casos contemplados consiste en la vuelta al reposo, ocurrirá en el momento en el que se alcance la celda final, es decir, cuando se complete el recorrido, o en caso de que exista algún problema y sea necesario parar. Además, en cualquier momento, desde el mando de control remoto podrá desactivarse el movimiento del rover. Esto facilitará que tenga lugar un movimiento seguro en última instancia.

Los bloques que hacen referencia a movimientos de giro y rectilíneo se llevan a cabo tal y como se ha explicado detalladamente en la sección previa, de acuerdo al algoritmo de control desarrollado con este objetivo.

Además, para poder realizar toda la comunicación de manera inmediata, y sin necesidad de tecnologías inalámbricas entre distintos ordenadores, se ha incorporado un microordenador a la estructura del rover, que será el encargado de ejecutar simultáneamente todos los flujos de programas de los diferentes nodos creados. Esto es posible gracias a la ejecución de todos los programas dentro del mismo espacio de trabajo, como nodos independientes, únicamente comunicándose mediante los conjuntos de datos necesarios para el funcionamiento global del sistema.

3.8. Resumen del Capítulo Algoritmo Propuesto

En el tercer capítulo de la memoria, se comienza exponiendo las reglas de la competición en la que participará el rover, de manera que se contextualiza el trabajo al mismo tiempo que se justifican las decisiones tomadas durante su desarrollo.

En segundo lugar, se explica cada uno de los componentes de hardware elegidos para el rover y se argumenta su elección entre todas las disponibles. Los componentes utilizados son, un UGV Rover Wild Thumper, con seis motores de corriente continua alimentados a 6V. Un dispositivo Pixhawk 2, que hará las funciones de control de nivel bajo, además de incorporar ciertos sensores utilizados durante el movimiento. ESC o variadores para motores con escobillas. Un microordenador Odroid XU4 cuya función será la de ejecutar el controlador implementado. Un sistema GPS Here+ con un dispositivo RTK que permitirá obtener medidas más precisas. Y, por último, dos baterías que alimentarán a todos los componentes del sistema.

Posteriormente, se detalla la elección de software realizada, tanto para la estación terrestre como para desarrollar el controlador. Se explica el proceso seguido de calibración de todos los componentes ensamblados en el chasis del vehículo, y las pruebas realizadas para comprobar su correcto funcionamiento de manera conjunta.

Para continuar, se justifica la elección de control óptimo para desarrollar el algoritmo de movimiento del rover. Además, se detalla el proceso seguido hasta obtener las ecuaciones que definen el movimiento que se desea realizar (3.35 y 3.42), traslaciones rectilíneas de 1 metro y giros estáticos que permitan al rover orientarse en los cuatro sentidos, todo ello para cruzar las celdas del terreno en el que se llevará a cabo la competición. Se incluyen también flujogramas que detallan el funcionamiento del programa que se ha realizado para que se lleve a cabo el comportamiento recogido en dichas ecuaciones. Se explican, asimismo, todas las librerías del entorno ROS implementadas que permiten enviar y recibir datos de todos los sensores y actuadores.

En última instancia, se detalla la comunicación que ha de realizar el controlador con otros sistemas durante la competición, tales como la recepción por parte del controlador de las indicaciones de dirección y sentido necesarias para el movimiento, y el envío de confirmaciones de llegada al destino.

4. ANÁLISIS DE RESULTADOS

Una vez se ha desarrollado y programado el algoritmo de control deseado para el movimiento autónomo del UGV, es necesario comenzar a realizar pruebas, que validen el planteamiento además de permitir llevar a cabo ciertos ajustes en el código realizado.

4.1. Pruebas realizadas

Debido a la naturaleza del terreno en el cual se desarrollará la competición, y el limitado terreno disponible en el campus de Leganés de la UC3M, se han realizado las pruebas ejecutando los movimientos de manera independiente y no consecutiva, excepto en casos puntuales. Para un mayor conocimiento de las fortalezas y limitaciones del controlador desarrollado para el UGV, se han realizado múltiples pruebas del movimiento rectilíneo en diferentes tipos de terrenos, cuyas diferencias residen en la regularidad de la superficie de estos y la adherencia de las ruedas a los mismos. Además, se ha probado cada uno de los tres giros que contempla el movimiento en diversas ocasiones. Por último, se han realizado test que ejecutan ambos movimientos de manera conjunta, para conocer su resultado y las limitaciones que implican.

Para llevar a cabo las pruebas, tanto aquellas que ejecutan secciones particulares del controlador, como las que lo hacen en su totalidad, es necesario acceder al microordenador Odroid anclado al rover de manera remota. Esto se llevará a cabo accediendo a su red Wifi (AP), y ejecutando comandos SSH mediante el terminal. Cabe destacar que las pruebas de secciones específicas del controlador se llevan a cabo evitando la ejecución de ciertas partes del código fuente de este, pero sin modificarlo. Sin embargo, en todos los casos se utilizará el envío de mensajes entre nodos, pero para que el controlador realice ciertas acciones particulares, dichos mensajes indicarán la realización de la acción que se está probando, bien sea un giro o un movimiento rectilíneo.

Es asimismo importante, tanto en la realización de las pruebas como en la competición, ejecutar una serie de comandos en la terminal del microordenador en un orden determinado, los mismos tendrán la función de lanzar ROS y MAVROS, ejecutar el software del piloto automático y todos los nodos. Dichos comandos son, `roscore`, encargado de ejecutar el entorno ROS en el ordenador. Posteriormente `roslaunch`

`mavros apm.launch`, encargado de ejecutar la librería MAVROS, y en última instancia han de ejecutarse todos los nodos que participen en la ejecución del código mediante `rosrun *nombre del paquete* *nombre del nodo*`.

4.2. Resultados de las pruebas

Con el objetivo de obtener la mayor precisión posible en los resultados de las pruebas realizadas, se han dividido las mismas en dos grupos, el primero de ellos consistirá en ejecutar el controlador que lleva a cabo el giro del vehículo en múltiples ocasiones, para así obtener tendencias en los resultados. En el caso particular del giro, es necesario realizar los tres tipos de giro, 90º a la derecha, 180º a la derecha, y 90º a la izquierda en todos los terrenos analizados. Mientras que, para el controlador del movimiento rectilíneo, únicamente se cambiará el terreno de pruebas.

4.2.1. Resultados de giros

En primer lugar, se han realizado los giros de 90º sentido horario, los mismos se han llevado a cabo en diferentes superficies, tanto baldosas pulidas, como adoquines u hormigón impreso y tierra. Los resultados obtenidos en las mismas dependen en gran medida de las lecturas que se lleven a cabo de la brújula interna de la Pixhawk, así como de la precisión de las ruedas. Dicha precisión depende en gran medida de la capacidad que tienen los ESC de proveer de suficiente corriente a todas las ruedas en el mismo instante. Teniendo esto en cuenta, se ha procedido a realizar los diferentes ensayos.

Los datos obtenidos del controlador para una muestra de los giros de 90º grados realizados en sentido horario se muestran en la tabla 6. En la tabla aparecen un total de 20 ensayos, los cuales se han recogido tras la calibración del controlador, pese a que el número total de pruebas realizadas ha sido mayor.

Pruebas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Ángulo girado (º)	88,92	88,9	89,88	91,47	92,15	91,36	89,35	87,89	89,17	91,56

11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
87,03	88,31	87,19	87,27	90,63	87,2	88,17	90,43	90,83	91,25

Tabla 6. Resultados de cada una de las 20 pruebas recogidas para el giro de 90º en sentido horario

Los resultados oscilan entre 87,89º y 92,15º en todos los terrenos ensayados, es decir, el error es en el peor de los casos de ±2º. Dicho dato confirma que las lecturas llevadas a cabo por la brújula interna del sistema FCU son adecuadas y que el control sobre los motores permite un giro preciso en distancias cortas.

Los resultados obtenidos en los ensayos se muestran en forma de gráfica a continuación, en ellos puede observarse con respecto a una línea de referencia que marca los 90º deseados en el giro la curva realizada por el rover sobre su eje z. El eje x de la gráfica muestra cada una de las pruebas, mientras que el eje y recoge la información del ángulo girado en cada ensayo.

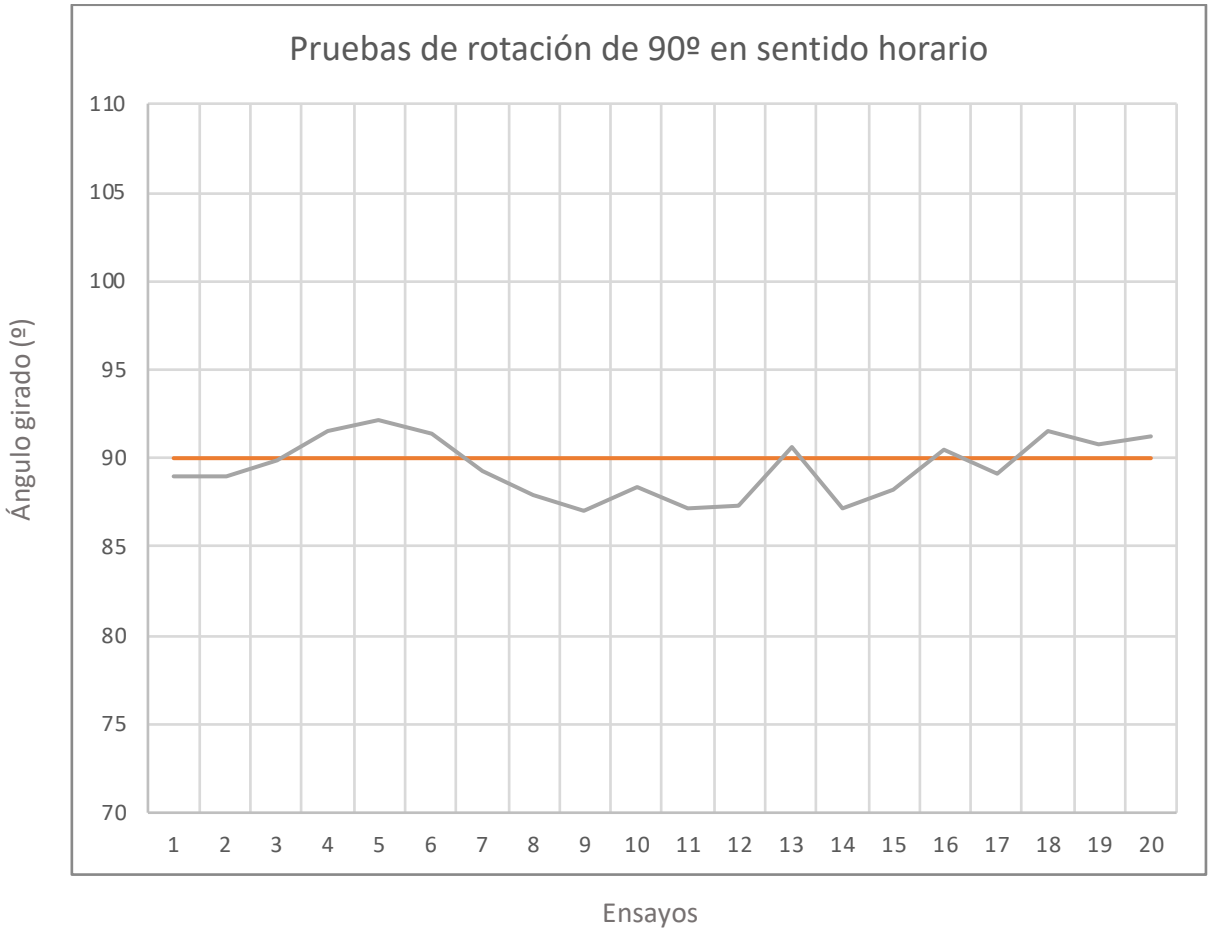

Fig. 4.1. Pruebas de giros estáticos de 90º en sentido horario

Es posible concluir que los resultados obtenidos para los giros, independientemente del terreno, ya que los ensayos mostrados en la gráfica corresponden a diferentes tipos de

pavimento, son totalmente precisos, ya permitirán al rover rotar hacia la derecha durante la competición sin acumular errores significativos de posición,

En el caso de los giros en sentido antihorario, de igual manera que en casos anteriores, se han realizado numerosos ensayos en diferentes terrenos, con mayor y menor adherencia y estabilidad. En un primer momento fue necesario calibrar el controlador para que trabajara de manera precisa en cualquiera de ellos. Una vez se logró este objetivo, comenzaron las pruebas que buscan dar validez al controlador, algunas de ellas se reflejan en la tabla 7, un total de 20. En ellas puede observarse que la tendencia es claramente a finalizar el movimiento en posiciones angulares relativas cercanas a los 90º.

Prueba	1	2	3	4	5	6	7	8	9	10
Ángulo girado (º)	89,78	91,34	88,14	89,12	90,02	87,56	90,92	91,43	89,47	91,79

11	12	13	14	15	16	17	18	19	20
91,37	92,07	87,14	89,58	87,76	90,95	92,02	90,74	89,02	88,21

Tabla 7. Resultados de cada una de las 20 pruebas recogidas para el giro de 90º en sentido antihorario

De nuevo en este caso, el error de movimiento oscilará en medidas de ±2º, con lo que, todos los giros en sentido antihorario, se realizarán de manera rigurosa, independientemente del terreno en el que tengan que realizarse. Tal y como ocurre en el caso anterior, el error acumulado tras la realización de múltiples maniobras será mínimo gracias a la elevada precisión.

En la gráfica 4.2, puede observarse que la tendencia es claramente a realizar rotaciones con un recorrido muy cercano al cuarto de circunferencia en todos los casos estudiados.

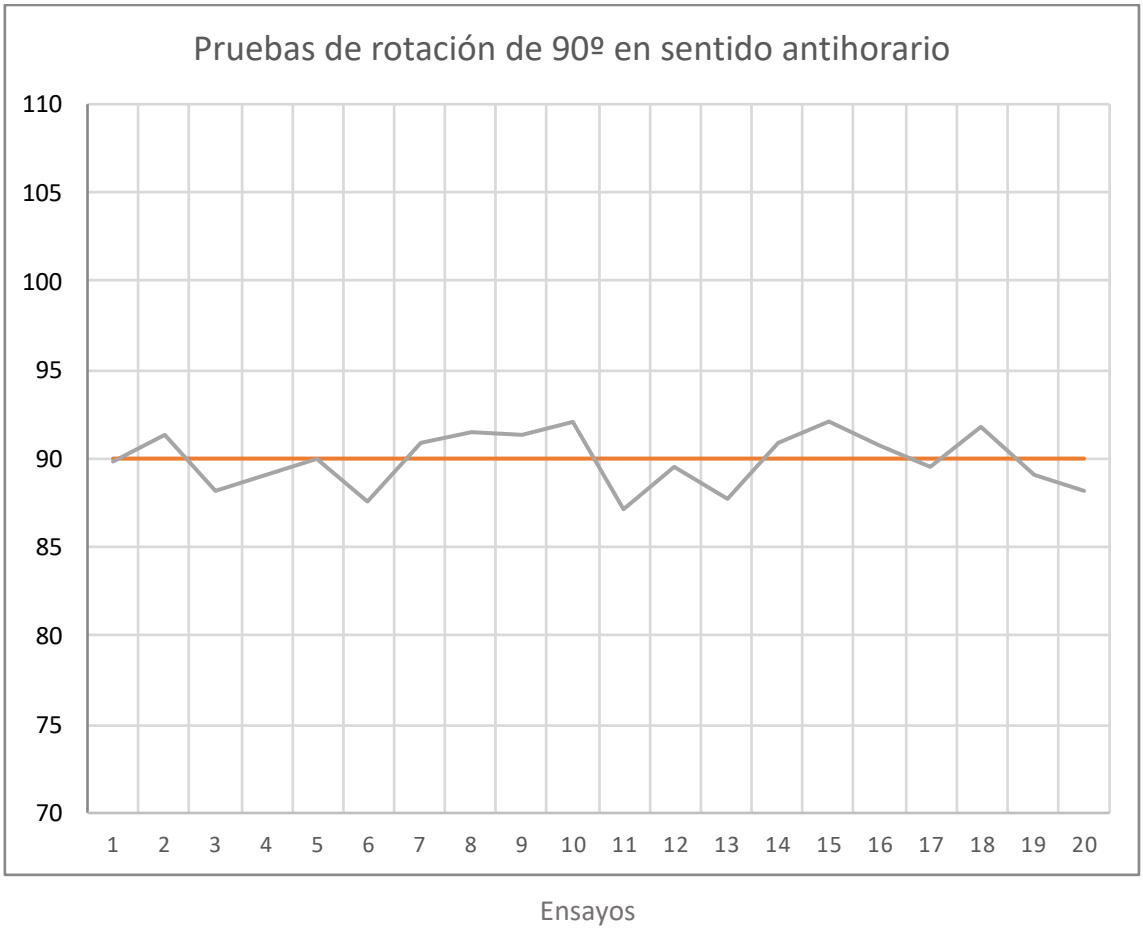

Fig. 4.2. Pruebas de giros estáticos de 90º en sentido antihorario

Para finalizar, el último giro que es necesario realizar, consiste en un cambio de sentido sin modificar la posición. Al igual que en casos previos, se ha realizado en múltiples terrenos, y tras una minuciosa calibración, se han obtenido los resultados mostrados en la tabla 8.

Prueba	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Ángulo girado (º)	177,9	176,87	178,76	178,47	178,46	180,04	178,4	177,9	179,48	181,92
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
178,7	177,07	179,18	178,1	183,41	180,77	182,12	181,58	178,95	181,21	

Tabla 8. Resultados de cada una de las 20 pruebas recogidas para el giro de 180º en sentido horario

Si se muestran los datos obtenidos en forma de gráfica, es posible observar las tendencias que siguen los datos recogidos, tal y como es el caso en la figura 4.3, para los giros de 180º.

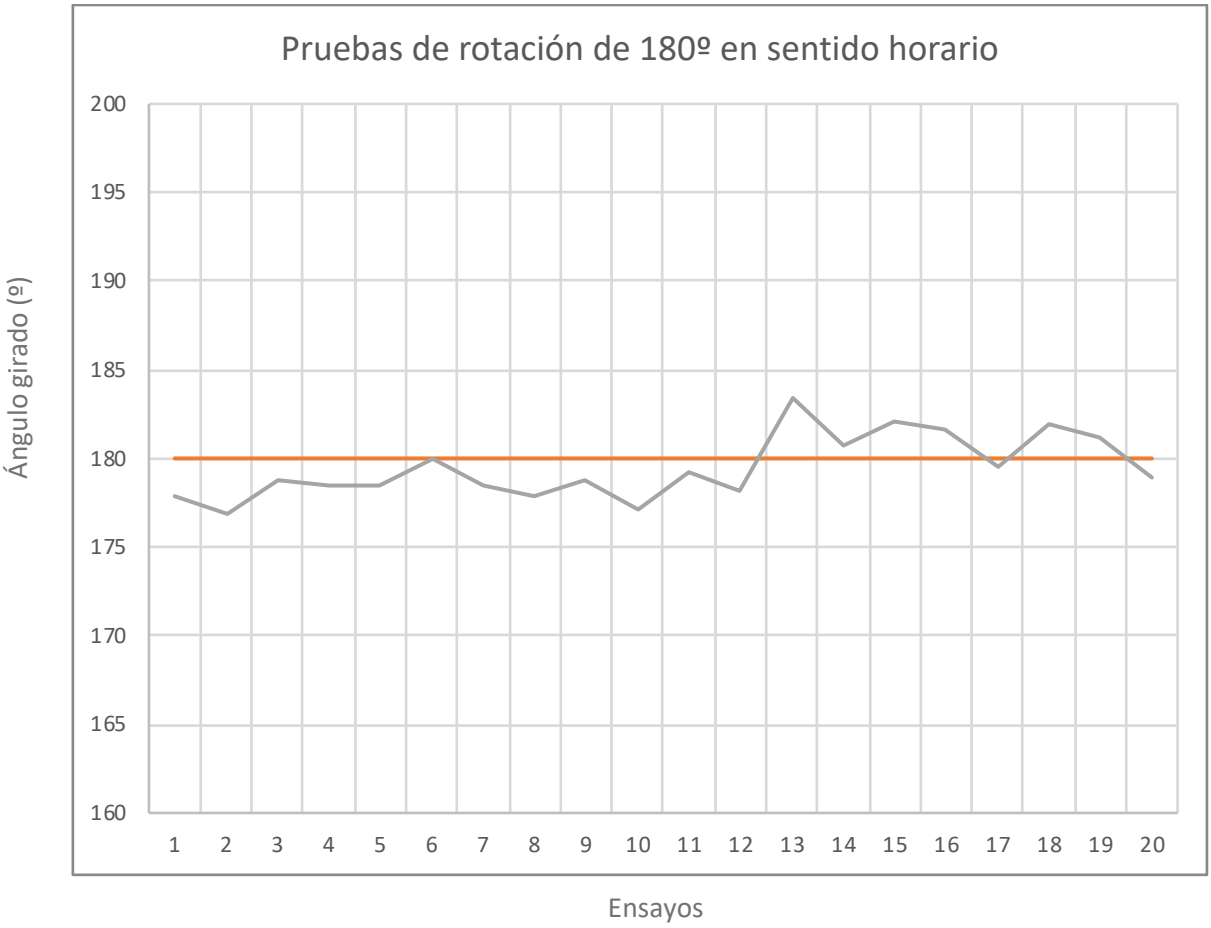

Fig. 4.3. Pruebas de giros estáticos de 180º en sentido horario

En este último caso, de nuevo se ha conseguido una tendencia que indica que todos y cada uno de los giros se realiza de forma que el error sea mínimo, oscilando entre los ±3º. Puede concluirse que, en este caso, la precisión es menor, debido a que el giro es mayor, así como la velocidad alcanzada por el UGV, sin embargo, será asimismo el movimiento que se repita en menos ocasiones.

4.2.2. Resultados de movimiento rectilíneo

Para probar el controlador que ejecuta el algoritmo basado en control óptimo se llevarán a cabo dos tipos de pruebas diferentes, la primera de ellas basará la posición en las lecturas recibidas del GPS sin hacer uso del módulo RTK, utilizando además las medidas obtenidas mediante odometría con el fin de que no se utilicen posibles lecturas del GPS muy dispares. Para las segundas pruebas se utilizará el módulo RTK, de esta manera

podrá comprobarse la precisión obtenida por ambas medidas y la necesariedad del uso del módulo RTK.

Los resultados obtenidos en los ensayos sin RTK se muestran en tres gráficas diferentes, cada una de ellas otorgando una ponderación diferente a la odometría. Esto se lleva a cabo en los puntos en los que la medición del GPS y la odometría difiere en más de un 20%, entonces se realiza una ponderación de ambas medidas, de tres maneras diferentes. La primera de ellas iguala el peso de ambas medidas, es decir, se calcula la media, mientras que, en las posteriores, la odometría tendrá un peso del 60 y 70% respectivamente.

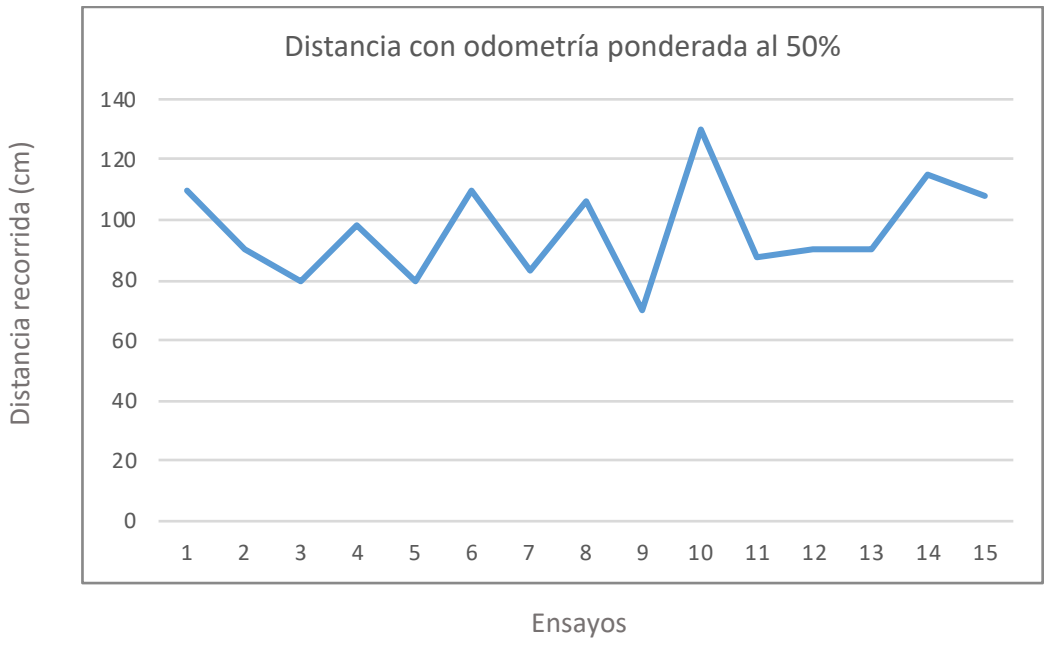

Fig. 4.4. Gráfica que muestra los datos de distancias recorridas con la odometría ponderada al 50% con el GPS

En el primer caso, cuando la odometría tiene el mismo peso en la medida que el GPS, puede observarse que los resultados obtenidos no se adecúan a las necesidades del movimiento, por ello, se realizan pruebas otorgando un mayor peso a la odometría. La media de las distancias recorridas es de alrededor de 92 cm, valor que pese a tener un error de un 8%, atendiéndose a los ensayos individuales, se observa que los resultados obtenidos distan en gran medida del objetivo de 100cm.

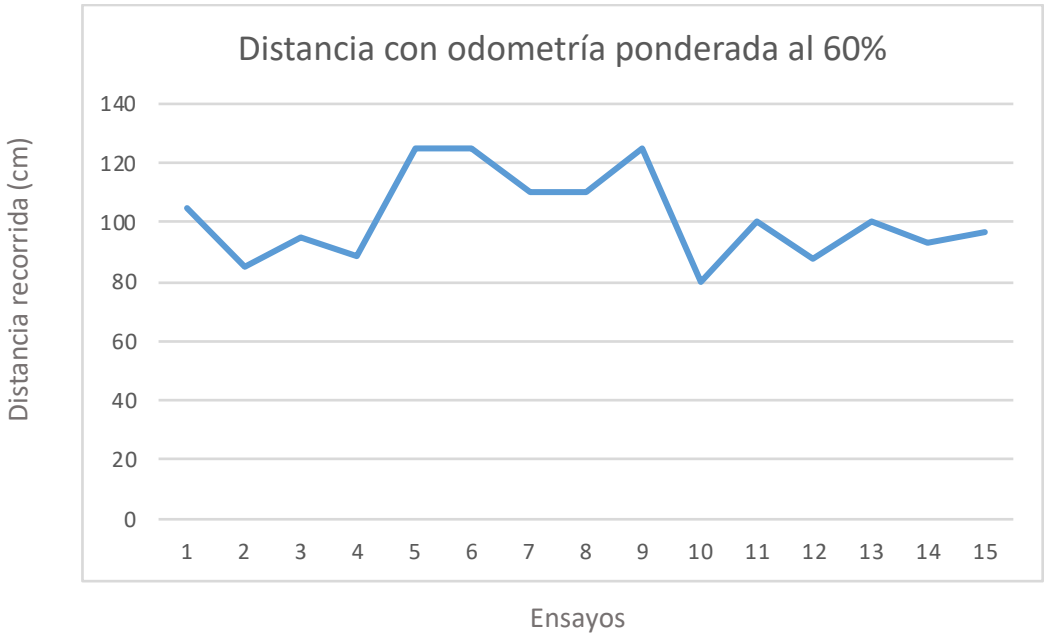

Fig. 4.5. Gráfica que muestra los datos de distancias recorridas con la odometría ponderada al 60% con el GPS

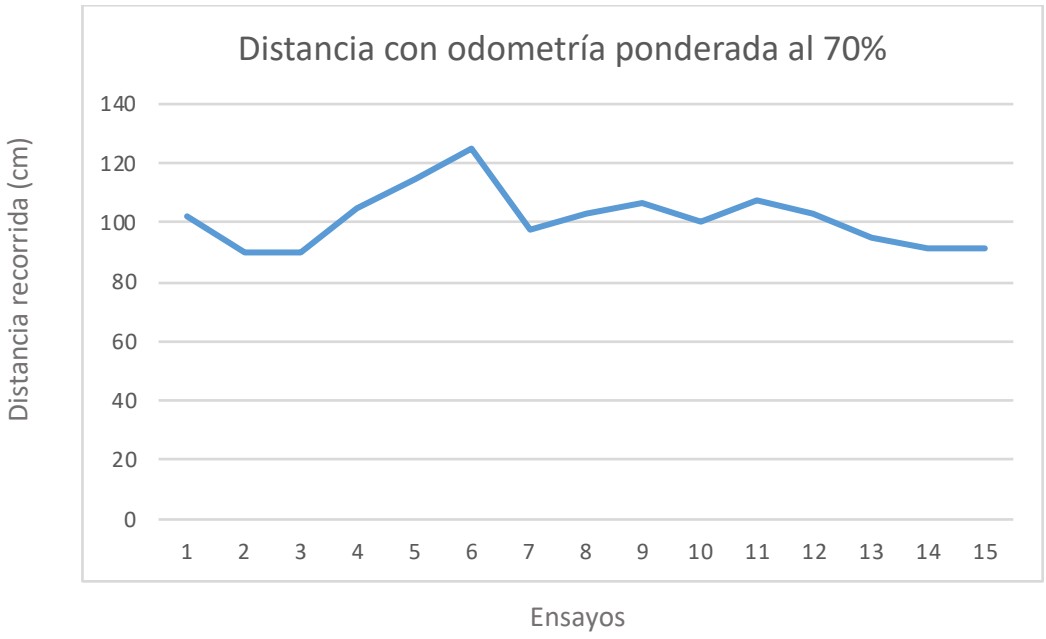

Fig. 4.6. Gráfica que muestra los datos de distancias recorridas con la odometría ponderada al 70% con el GPS

En los casos en los que se otorga un mayor valor a la odometría, se obtienen resultados más similares a los deseados, aceptables. La distancia media recorrida en todas las pruebas es de 101,8 cm y 101,53 cm respectivamente. Dichos datos, y los procedimientos utilizados para obtenerlos, hacen patente el hecho de que el GPS no proporciona las

mediadas adecuadas de precisión para esta aplicación. Esto es así, ya que la odometría, pese a ser un método de medida bastante utilizado, carece de una alta precisión, ya que no tiene en cuenta factores externos del terreno, tales como pendientes o hendiduras, todo el movimiento lo considera en el plano. Dado que se observa que la odometría procura medidas más precisas que el GPS, puede concluirse que el uso de GPS sin RTK no es válido en esta aplicación.

En el momento en el que se incorpora el sistema RTK, se observa que el resultado obtenido mejora notablemente. En la tabla 9 se muestran algunas de las distancias medidas que ha recorrido el rover. Dichas distancias se han calculado únicamente utilizando las medidas obtenidas del GPS, sin incluir los resultados obtenidos por odometría, ya que se trata de un método de cálculo poco preciso.

Prueba	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Distancia(cm)	104	103	104	99	104	102	103	98	101	99	104	98	102	99	103

Tabla 9. Resultados obtenidos del movimiento rectilíneo del UGV utilizando únicamente el dispositivo GPS con el módulo RTK

Los resultados obtenidos pueden considerarse totalmente satisfactorios, ya que la mayor desviación que se observa es de 4cm, es decir un 4% del recorrido total. Esto puede observarse claramente en la gráfica 4.7, donde se presenta la tendencia seguida por el UGV al realizar las pruebas.

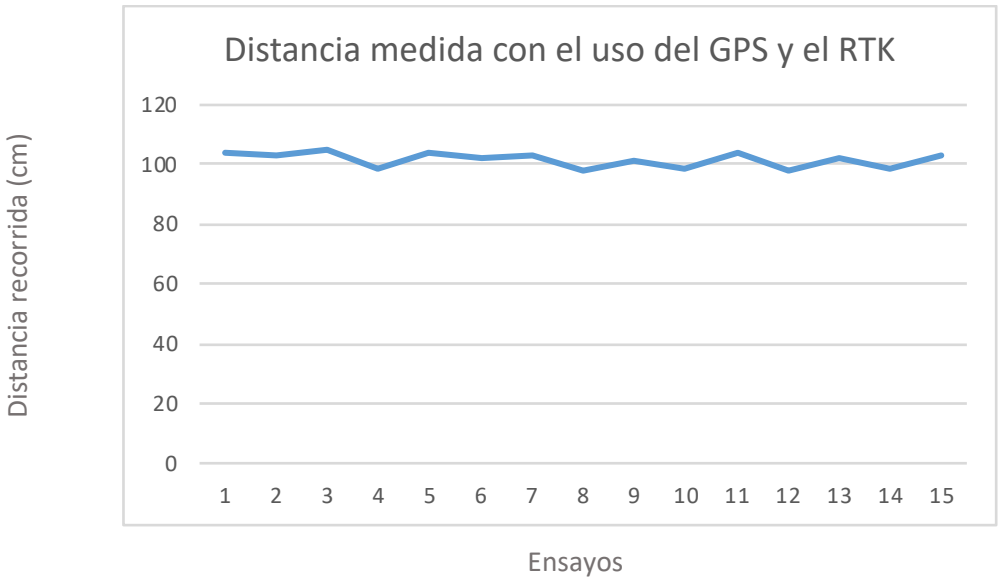

Fig. 4.7. Gráfica que muestra los resultados obtenidos del movimiento rectilíneo del UGV utilizando únicamente el dispositivo GPS con el módulo RTK

Pese a basar los cálculos de velocidad y posición únicamente en los valores obtenidos por el GPS, se obtienen unos resultados que validan el uso del controlador en la competición, ya que la variación de las medidas obtenidas es mínima.

Por otro lado, si se incluyen los datos obtenidos por odometría en la estimación de la posición del rover, ponderando ambos datos al 50% en los casos en los que la medida difiera más de un 20%, se observa que la precisión del movimiento disminuye considerablemente, de la manera que se muestra en la tabla 10.

Prueba	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Distancia(cm)	104	107	98	108	117	89	109	98	103	107	105	103	95	103	96

Tabla 10. Resultados obtenidos del movimiento rectilíneo del UGV utilizando la odometría y el dispositivo GPS con el módulo RTK

Este comportamiento indica, tal y como se buscaba, que mediante el GPS se obtiene una lectura de la posición mucho más precisa que con la odometría. Este aspecto confirma que la odometría no es un método particularmente preciso en el que pueda basarse un controlador de posición, sin embargo, puede utilizarse como medida complementaria si el sistema principal carece de precisión, o si no esta no es necesaria. En la gráfica 4.8 puede observarse que en el caso de utilizar la odometría las medidas obtenidas acumulan un error de medida de alrededor del 10%, sustancialmente superior al obtenido en los casos en los que no se utiliza la odometría.

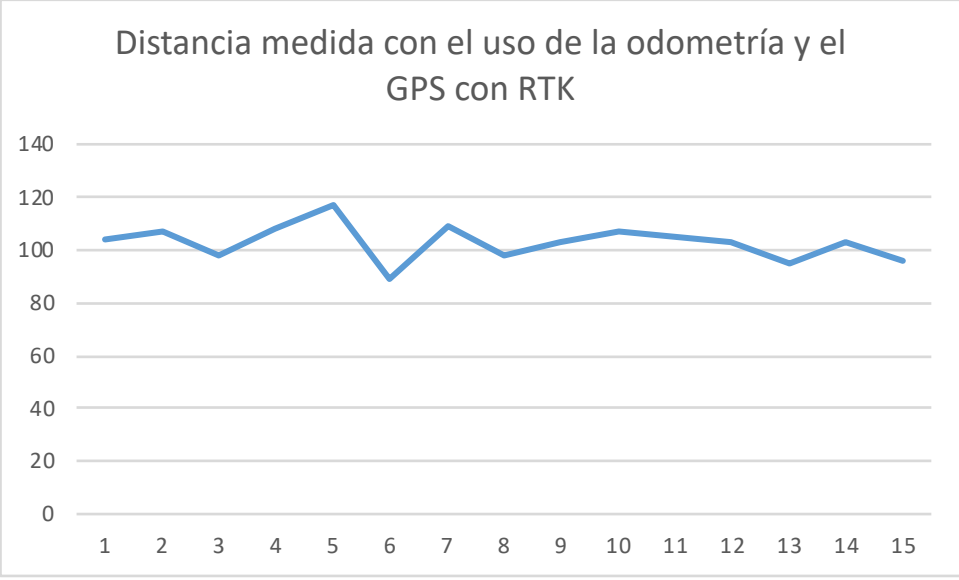

Fig. 4.8. Gráfica que muestra los resultados obtenidos del movimiento rectilíneo del UGV utilizando la odometría y el dispositivo GPS con el módulo RTK

Puede concluirse, que la configuración elegida es aquella que hace uso del GPS con el módulo RTK, sin utilizar la odometría como método complementario, ya que arroja los resultados más precisos y fiables.

4.2.3. Resultados de pruebas conjuntas

En el caso de las pruebas conjuntas, no se han incluido resultados de nuevo, ya que se repiten los patrones mostrados en las secciones previas, tanto en los giros como en los movimientos en línea recta. En el primer caso con una elevada precisión, y un error acumulado irrelevante, mientras que las distancias recorridas acumulan un error mínimo, del orden del 4%.

4.3. Resumen del Capítulo Análisis de Resultados

En el presente capítulo se han expuesto los resultados de todos los ensayos y pruebas realizados del controlador del vehículo.

En primera instancia se explican todos los terrenos utilizados para llevar a cabo las pruebas, todos ellos con diferente agarre y estabilidad, entre ellos tierra, baldosas y hormigón impreso o adoquines.

Las primeras pruebas realizadas llevan a cabo únicamente la ejecución de la parte del control dedicada al giro, en sus tres variantes, 90º en sentido horario y antihorario, y 180º en sentido horario. Este primer grupo de pruebas arroja datos que permiten concluir que el giro se realizará de manera precisa, obteniéndose un error máximo de alrededor de ±3º en los casos más desfavorables.

Posteriormente se realizan ensayos para el fragmento del controlador que ejecuta el movimiento rectilíneo, en primera instancia utilizando únicamente el GPS sin el módulo RTK, y posteriormente incluyendo este último. Las pruebas llevadas a cabo sin el RTK carecen de la precisión necesaria para su validez, sin embargo, en el momento en el que se incluye el RTK, la precisión del GPS aumenta considerablemente y se obtienen resultados que son totalmente válidos para su uso durante la competición.

5. CONCLUSIONES

A lo largo del trabajo se ha expuesto el proceso completo de desarrollo e implementación de un controlador para el movimiento de un vehículo autónomo diferencial. Se han abordado temas tales como la elección de hardware y software, y se ha documentado y justificado un riguroso proceso de pruebas para la validación del controlador.

Teniendo todo esto en cuenta, puede concluirse el trabajo afirmando que, pese a su reducido uso en aplicaciones de ingeniería, el control óptimo ha resultado ser un método extremadamente útil y sencillo de implementar para el control del movimiento autónomo. Con él se han obtenido unos resultados remarcables, pese a que la aplicación es especialmente compleja, ya que las distancias recorridas son muy reducidas, y las medidas de sensores conllevan errores relativos elevados.

Es asimismo reseñable la integración de todos los elementos de hardware escogidos para el proyecto con el entorno de software ROS, ya que todos los elementos han trabajado de manera conjunta, intercambiando datos de manera instantánea, pese a tratarse de dispositivos de diferentes fabricantes y áreas de aplicación.

Para terminar, cabe destacar la factibilidad de la extrapolación de este tipo de control a otras aplicaciones más complejas, tales como la conducción de vehículos autónomos. Con la correspondiente implementación de sensores y actuadores acordes a la aplicación escogida.

5.1. Trabajos futuros

Pese a que los resultados obtenidos son plenamente satisfactorios, como en la mayoría de los proyectos de ingeniería, siembre pueden realizarse mejoras. En este caso, la principal flaqueza del controlador es sin duda la estimación de la posición. Por ello, y para futuras competiciones, una posible alternativa al uso del GPS es la implementación de enconders o codificadores, que permitan estimar la posición de una manera mucho más precisa. Ya que, sin incurrir en costes más elevados que los del propio, se conseguiría un control mucho más preciso, especialmente si se incluye uno en cada una de las seis ruedas del vehículo.

Además, el otro elemento hardware que mejoraría la ejecución del controlador son los motores, dado que se trata del elemento básico que ejerce el movimiento, y que, utilizándose modelos con un menor tiempo de respuesta y una mayor sensibilidad, se obtendrían resultados más precisos, pero en un menor espacio de tiempo. Todo ello utilizando los ESC adecuados para los motores elegidos, siendo capaces de proveer a estos de los valores de voltaje y corriente necesarios en cada caso, en el menor tiempo posible.

En definitiva, en caso de llevarse a cabo las mejoras mencionadas, se obtendría un vehículo igualmente autónomo, pero reduciendo drásticamente los posibles errores cometidos en el control gracias a componentes más precisos que mejorarían el comportamiento general del UGV.

BIBLIOGRAFÍA

[1] «Wild Thumper 6WD Multi Chassis,» [En línea]. Available: https://www.sparkfun.com/products/11056. [Último acceso: 2 Agosto 2018].

[2] D. W. Gage, «A Brief History of Unmanned Ground Vehicle (UGV) Developement Efforts,» *Unmanned Systems Magazine,* vol. 13, nº 3, 1995.

[3] M. Rovery, «BAE Systems develops advanced UGV,» 05 09 2017. [En línea]. Available: https://www.janes.com/article/73554/bae-systems-develops-advanced-ugv. [Último acceso: 6 Agosto 2018].

[4] L. Ványa, «Excepts from the history of unmanned ground vehicles developements in the USA.,» *AARMS,* vol. 2, nº 2, pp. 185-197, 2003.

[5] E. Howell, «A Brief History of Mars Missions,» 17 Marzo 2015. [En línea]. Available: https://www.space.com/13558-historic-mars-missions.html. [Último acceso: 6 Agosto 2018].

[6] E. Howell, «Mars Curiosity: Facts and Information,» 17 Julio 2018. [En línea]. Available: https://www.space.com/17963-mars-curiosity.html. [Último acceso: 6 Agosto 2018].

[7] M. J. Roman, "Design and Analysis of a Four Wheeled Planetary Rover", M.S. thesis, University of Oklahoma, Norman, OK, USA, 2005. [En línea]. Available: http://c3p0.ou.edu/IRL/Theses/Roman-MS.pdf.

[8] M. Alberty, «Rover Searches California Desert for Water to Simulate Future Lunar Missions,» 16 Octubre 2014. [En línea]. Available: https://www.nasa.gov/ames/rover-searches-california-desert-for-water-to-simulate-future-lunar-missions/. [Último acceso: 6 Agosto 2018].

[9] G. S. Vasilash, «How to Engineer a Planetary Rover,» Automotive Design&Production, 14 Agosto 2012. [En línea]. Available: https://www.adandp.media/blog/post/how-to-engineer-a-planetary-rover. [Último acceso: 7 Agosto 2018].

[10] I. Chipman, «Hedgehog, a guide to asteroids and things that whirl around the solar system,» *Stanford Engineering Magazine,* 2016. [En línea]. Available:

https://engineering.stanford.edu/magazine/article/hedgehog-guide-asteroids-and-things-whirl-around-solar-system.

[11] N. Shoemaker, «Hedgehog rover will be our space guides to asteroids,» 2 Septiembre 2016. [En línea]. Available: https://www.geek.com/news/hedgehog-rover-will-be-our-space-guides-to-asteroids-1646807/. [Último acceso: 7 Agosto 2018].

[12] A. B. Azcón, "Análisis y Diseño del Control de Posición de un Robot Móvil con Tracción Diferencial", Proyecto de final de carrera, Universitat Rovira I Virgili, Tarragona, 2003. [En línea]. Available: http://deeea.urv.cat/public/PROPOSTES/pub/pdf/333pub.pdf.

[13] N. E. Sánchez, "Diseño de la arquitectura funcional de control para un robot móvil de exploración espacial", Trabajo de final de grado, Universidad Carlos III de Madrid, Madrid, 2012. [En línea]. Available: https://e-archivo.uc3m.es/handle/10016/16891.

[14] A. M. Ortiz, "ANÁLISIS DEL COMPORTAMIENTO DINÁMICO DE UN VEHÍCULO CON CONTROLADOR PID", Proyecto final de carrera, Universidad Carlos III de Madrid, Madrid, 2014. [En línea]. Available: https://e-archivo.uc3m.es/handle/10016/22558.

[15] J. Villagrá, V. Milanés, J. Pérez y T. d. Pedro, «Control basado en PID inteligentes: aplicación al control de crucero de un vehículo a bajas velocidades,» *Revista Iberoamericana de Automática e Informática Industrial,* vol. 7, nº 4, pp. 44-52, octubre 2010.

[16] F. V. P. Ramos, "Diseño y Comprobación de un Módulo para Control Óptimo de Posición de un Motor D.C.", Tesis de grado previa a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones, Escuela Politécnica Nacional, Quito., 1983. [En línea]. Available: http://bibdigital.epn.edu.ec/handle/15000/11393.

[17] PX4, «Cube Flight Controller,» [En línea]. Available: https://docs.px4.io/en/flight_controller/pixhawk-2.html. [Último acceso: 10 Agosto 2018].

[18] Open Surce Robotics Foundation, «About ROS,» [En línea]. Available: http://www.ros.org/about-ros/. [Último acceso: 10 Agosto 2018].

[19] J. C. Fernández, "DISEÑO DE UN SISTEMA DE CONTROL PARA UN CUADRICÓPTERO", Trabajo fin de grado,Departamento de Ingeniería de Sistemas y Automática, Universidad

Carlos III de Madrid, Madrid, España, 2013. [En línea]. Available: https://e-archivo.uc3m.es/handle/10016/18750.

[20] The Construct, «ROS Tutorials | Intro to Robot Operating System,» 22 Septiembre 2017. [En línea]. Available: http://www.theconstructsim.com/ros-for-beginners/. [Último acceso: 10 Agosto 2018].

[21] A. Mahtani, L. Sánchez, E. Fernández y A. Martinez, «ROS Architecture and Concepts,» 28 Diciembre 2016. [En línea]. Available: https://hub.packtpub.com/ros-architecture-and-concepts/. [Último acceso: 10 Agosto 2018].

[22] «Minesweepers Academia,» [En línea]. Available: http://www.landminefree.org/minesweepers-academia/. [Último acceso: 13 Agosto 2018].

[23] «Variador Hobbywing QUICRUN 1060 Brushed,» [En línea]. Available: http://www.rcmitico.es/es/productos-hobbywing/variador-hobbywing-quicrun-1060-brushed. [Último acceso: 16 Agosto 2018].

[24] «Cube Flight Controller,» [En línea]. Available: https://docs.px4.io/en/flight_controller/pixhawk-2.html. [Último acceso: 16 Agosto 2018].

[25] «Here+ V2 RTK GNSS (M8P),» [En línea]. Available: http://www.proficnc.com/system-kits/77-gps-module.html. [Último acceso: 16 Agosto 2018].

[26] «ODROID-XU4,» [En línea]. Available: https://www.hardkernel.com/main/products/prdt_info.php. [Último acceso: 16 Agosto 2018].

[27] «Turnigy 9X 9Ch Transmitter w/ Module & iA8 Receiver,» [En línea]. Available: https://hobbyking.com/es_es/turnigy-9x-9ch-transmitter-w-module-ia8-receiver-mode-2-afdhs-2a-system.html. [Último acceso: 2018 Agosto 2018].

[28] «Flight Controller/Sensor Orientation,» [En línea]. Available: https://docs.px4.io/en/config/flight_controller_orientation.html. [Último acceso: 22 Agosto 2018].

ANEXO A. INSTALACIÓN DEL SOFTWARE REQUERIDO.

Para llevar a cabo la instalación del sistema operativo ROS utilizado en el presente trabajo, es necesario utilizar la terminal de Linux, así como una serie de comandos que permitirán a Ubuntu acceder al repositorio en el que se encuentran los paquetes de software necesarios.

El primer paso consiste en configurar el ordenador para aceptar software de packages.ros.org. Esto se lleva a cabo mediante el comando:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Posteriormente, se procede a preparar el sistema para aceptar *"keys"* relacionadas con ROS, para ello es necesario ejecutar el comando:

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 -
-recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

En este momento el SO está listo para comenzar con la instalación de ROS. Se continuará actualizando los paquetes que contienen los índices mediante el comando siguiente:

```
$ sudo apt-get update
```

En este momento ha de elegirse el tipo de instalación de ROS que queremos realizar. Se escoge en función de las librerías que se vayan a necesitar. En este caso particular, se ha elegido la instalación completa.

```
$ sudo apt-get install ros-kinetic-desktop-full
```

Luego se buscan los paquetes disponibles para la instalación de ROS escogida.

```
$ apt-cache search ros-kinetic
```

Una vez se ha hecho todo esto, ROS ha quedado completamente instalado, sin embargo, para poder utilizarlo es necesario inicializarlo y posteriormente configurar su entorno, para ello han de ejecutarse los siguientes comandos en el orden mostrado.

```
$ sudo rosdep init
$ rosdep update
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

En este momento es posible comenzar a usar ROS ya que la instalación está completa, sin embargo, para este trabajo es necesario además instalar MAVROS, para ello es necesario ejecutar de nuevo una serie de comandos en la terminal. El primer conjunto de comandos tiene la función de informar al instalador la ubicación del espacio de trabajo en el que se desea instalar MAVROS.

```
$ mkdir -p ~/*nombre_espacio_de_trabajo*/src
$ cd ~/*nombre_espacio_de_trabajo*
$ catkin init
$ wstool init src
```

Posteriormente ha de instalarse wstool, mediante

```
$ wstool init ~/catkin_ws/src
```

En este momento es necesario instalar el nodo de comunicación MAVLink.

```
$ rosinstall_generator --rosdistro kinetic mavlink | tee
/tmp/mavros.rosinstall
```

Ahora se realiza la misma acción con MAVROS.

```
$ rosinstall_generator --upstream mavros | tee -a
/tmp/mavros.rosinstall
```

Por último, es necesario compilar todos los elementos instalados para su posterior utilización. Para ello han de ejecutarse los siguientes comandos.

```
$ wstool merge -t src /tmp/mavros.rosinstall
$ wstool update -t src -j4
$ rosdep install --from-paths src --ignore-src -y
$ ./src/mavros/mavros/scripts/install_geographiclib_datasets.sh
$ catkin build
$ source devel/setup.bash
```

En este momento se ha completado la instalación de ROS y MAVROS y ambos se encuentran disponibles para su uso en el espacio de trabajo del usuario. Además, dado que se ha escogido una instalación completa de ambos, todos sus componentes están disponibles para su inmediata utilización.